

# Universidad de Alcalá

## Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

### Trabajo Fin de Grado

Desarrollo de funciones basadas en IP sobre el uP LPC1768

**Autor:** Álvaro Rosa Fernández

**Tutor/es:** Eliseo García garcía

2016

# UNIVERSIDAD DE ALCALÁ

## Escuela Politécnica Superior

Grado en Ingeniería Electrónica y Automática Industrial

### Trabajo Fin de Grado

Desarrollo de funciones basadas en IP sobre el uP LPC1768

**AUTOR:** Álvaro Rosa Fernández

**TUTOR:** Eliseo García García

#### TRIBUNAL

**Presidente:** Manuel Prieto Mateo

**Vocal 1º:** Juan Ignacio Pérez Sanz

**Vocal 2º:** Eliseo García García

**Fecha:** 30/09/2016

## ÍNDICE

1. Resumen en castellano .....	4
2. Resumen en inglés .....	5
3. Palabras clave .....	6
4. Introducción.....	7
5. Conceptos previos.....	8
5.1 Mundo Web .....	8
5.2 World Wide Web.....	10
5.3 HTML .....	13
5.4 Servidor.....	14
5.5 Microcontrolador .....	15
6. Objetivos y campo de aplicación .....	17
7. Descripción del proyecto .....	19
8. Requerimientos técnicos .....	20
8.1 Requerimientos Hardware.....	20
8.2 Requerimientos Software .....	40
9. Esquema del conexionado.....	45
10. Memoria .....	47
10.1 Análisis del programa .....	47
10.2 Desarrollo de funcionalidades .....	50
11. Conclusiones .....	58
12. Planificación temporal.....	59
13. Bibliografía.....	62
14. Anexos .....	63
14.1 Anexo 1. Código Easyweb original.....	63
14.2 Anexo 2. Código Easyweb definitivo .....	70

## 1. Resumen en castellano

Este proyecto trata de profundizar en la utilización de un microcontrolador como servidor web, basándonos en un ejemplo provisto por easyweb.

Una vez comprendido como lograr la correcta colaboración entre servidor e interfaz web, se tratará de desarrollar una funcionalidad que nos permita traducir variables medibles en el mundo real, en gráficas que se puedan apreciar a través de la interfaz.

De esta forma logramos observar de una forma interactiva que la comunicación se establece sin problemas y en tiempo real, a la vez que dejamos el camino abierto a futuras investigaciones que desarrollen funcionalidades más complejas basadas en este u otros microcontroladores.

## 2. Resumen en inglés

The goal for this Project is to investigate the possibility of using a microcontroller as a host for a web page.

We are going to develop an application that will analyze in real time a measurable magnitude and it will show up as a graph on the web page.

This way, not only we will program an application that can be used as a measure of things, but also it will open the path for future developers to investigate further on this field and develop better and more complex features using this idea of having a microcontroller as a host.

### 3. Palabras clave

Microcontrolador, servidor, interfaz, página web, aplicación

## 4. Introducción

En la época en la que nos encontramos, el avance de la tecnología está siendo muy rápido, y hoy en día es posible realizar aplicaciones de complejidad moderada en microcontroladores de bajo coste.

Por otro lado, tenemos en la actualidad una tecnología muy consolidada, como es el mundo Web, que mediante servidores web puede proporcionar información, que puede ir desde información de gran alcance como el estado de pequeños sensores particulares.

Es por este hecho por lo que queremos profundizar en el campo de la utilización de microcontroladores como servidores de páginas web. Es una opción sencilla y barata que nos permite comunicar, a través de Internet, toda la capacidad de análisis y matemática de la que goza un microcontrolador moderno, con la interfaz visual e inmediata que nos ofrece una página web.

El proyecto consiste, basándonos en un ejemplo de programa proporcionado por easyweb, en analizar la estructura necesaria para realizar esta conexión entre microcontrolador y web, para una vez comprendida, implementar alguna aplicación que nos permita comprobar que la conexión está teniendo lugar de forma satisfactoria y en el momento de la ejecución.

Este proyecto engloba un gran conjunto de materias debido a que requiere nociones de programación HTML y javascript, así como conocimientos de creación de servidores, o programación en C de pequeños dispositivos de bajo coste, basándonos en un microcontrolador concreto.

Es por esto por lo que considero este un proyecto muy completo y que abre las puertas a cualquier investigador que reúna las cualidades necesarias para investigar más a fondo el sinfín de aplicaciones que se pueden desarrollar utilizando este u otro microcontrolador similar, con un coste total muy reducido.

## 5. Conceptos previos

Antes de comenzar a explicar cualquier aspecto relacionado con el proyecto, es necesario realizar una breve descripción de ciertos conceptos clave que aparecerán mencionados varias veces a través de la memoria y cuyo entendimiento es base para la comprensión del trabajo.

Como ya hemos comentado en la introducción, este proyecto se encuentra ampliamente relacionado con el mundo web, sin embargo, ¿a qué nos referimos exactamente con mundo web?

### 5.1 Mundo Web

Es muy común confundir el origen de las páginas web con la creación de la red que a día de hoy conocemos por el nombre de Internet. Sin embargo, la aparición de ambas dista en el tiempo de más de una década y aunque tienen una fuerte correlación, sus orígenes son muy dispares.

El origen de Internet data del año 1969, cuando se estableció la primera conexión de computadoras entre tres universidades de California. Esta red que creó la Agencia para los Proyectos de Investigación Avanzada de los Estados Unidos se denominaba por aquel entonces ARPANET.



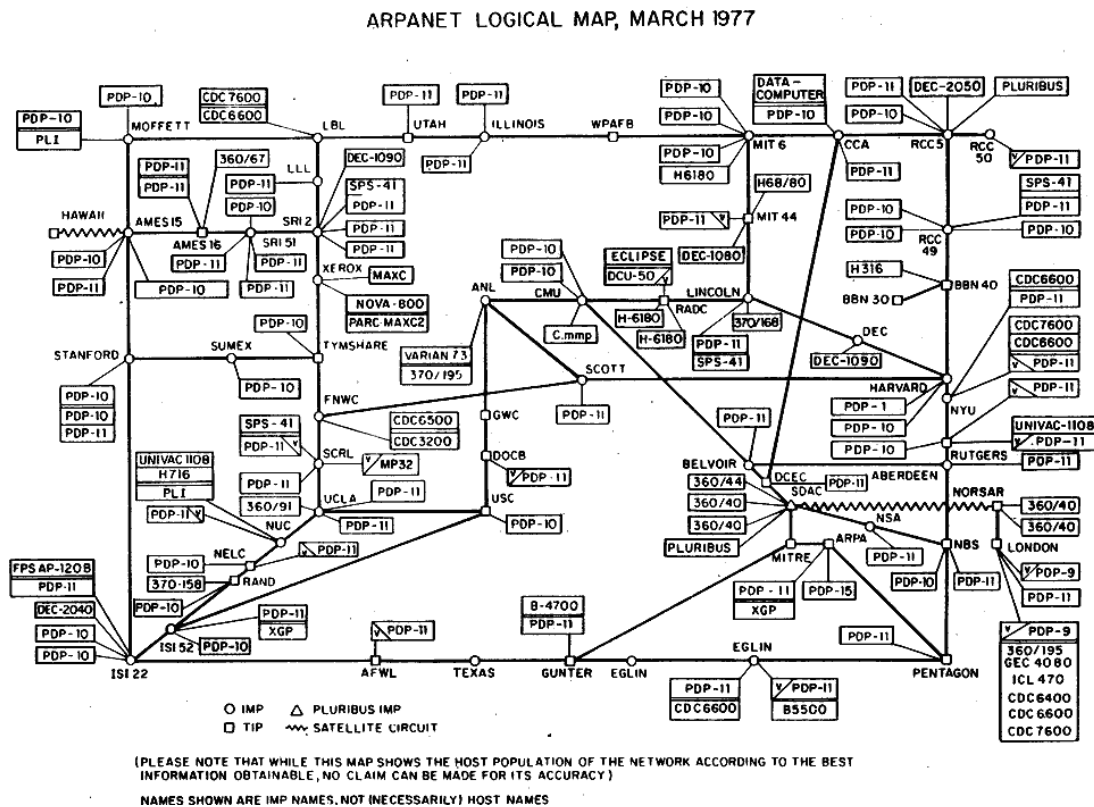


Figura 1. Esquema lógico de ARPANET

Esta red surge de la necesidad del gobierno estadounidense de protegerse frente a fallos no previstos en la red y debido a que los nodos de conmutación eran poco fiables.

No fue hasta dos décadas más tarde (1989) que un grupo de físicos dirigidos por Tim-Berners-Lee creó el lenguaje HTML que más tarde dio lugar a lo que hoy conocemos como World Wide Web.

## 5.2 World Wide Web

La World Wide Web (WWW) o red informática mundial es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces.



*Figura 2. Imagen de World Wide Web*

La creación de este entramado se puede atribuir (al igual que el código HTML) a Tim Berners-Lee, que, ya como personal de la división DD del CERN, redactó la propuesta que referenciaba a ENQUIRE y describía un sistema de gestión de información más elaborado. No hubo un bautizo oficial o un acuñamiento del término Web en esas referencias iniciales, utilizándose para tal efecto el término mesh. Sin embargo, el World Wide Web ya había nacido. Con la ayuda de Robert Cailliau, se publicó una propuesta más formal para la World Wide Web<sup>13</sup> el 6 de agosto de 1991.

Berners-Lee usó un NeXTcube como el primer servidor web del mundo y también escribió el primer navegador web, WorldWideWeb en 1991. En las Navidades del mismo año, Berners-Lee había creado todas las herramientas necesarias para que una web funcionase:<sup>14</sup> el primer navegador web (el cual también era un editor web), el primer servidor web y las primeras páginas web<sup>15</sup> que al mismo tiempo describían el proyecto.



*Figura 3. NextCube utilizado por Berners-Lee*

El 6 de agosto de 1991, envió un pequeño resumen del proyecto World Wide Web al newsgroup. Esta fecha también señala el debut de la Web como un servicio disponible públicamente en Internet.

El gran avance de Berners-Lee fue unir hipertexto e Internet. En su libro *Weaving the Web* (en castellano, *Tejiendo la Red*), explica que él había sugerido repetidamente que la unión entre las dos tecnologías era posible para miembros de las dos comunidades tecnológicas, pero como nadie aceptó su invitación, decidió, finalmente, hacer frente al proyecto él mismo.

World Wide Web tenía algunas diferencias de los otros sistemas de hipertexto que estaban disponibles en aquel momento:

- WWW solo requería enlaces unidireccionales en vez de los bidireccionales. Esto hacía posible que una persona enlazara a otro recurso sin necesidad de ninguna acción del propietario de ese recurso. Con ello se reducía significativamente la dificultad de implementar servidores web y navegadores (en comparación con los sistemas anteriores), pero en cambio presentaba el problema crónico de los enlaces rotos.

- A diferencia de sus predecesores, como HyperCard, World Wide Web era no-propietario, haciendo posible desarrollar servidores y clientes independientemente y añadir extensiones sin restricciones de licencia.

Respecto al funcionamiento de la web podemos añadir que el primer paso consiste en traducir la parte nombre del servidor de la URL en una dirección IP usando la base de datos distribuida de Internet conocida como DNS. Esta dirección IP es necesaria para contactar con el servidor web y poder enviarle paquetes de datos.



*Figura 4. Ejemplo de URL*

El siguiente paso es enviar una petición HTTP al servidor web solicitando el recurso. En el caso de una página web típica, primero se solicita el texto HTML y luego es inmediatamente analizado por el navegador, el cual, después, hace peticiones adicionales para los gráficos y otros ficheros que formen parte de la página. Las estadísticas de popularidad de un sitio web normalmente están basadas en el número de páginas vistas o las peticiones de servidor asociadas, o peticiones de fichero, que tienen lugar.

Al recibir los ficheros solicitados desde el servidor web, el navegador representa (renderiza) la página tal y como se describe en el código HTML, el CSS y otros lenguajes web. Al final se incorporan las imágenes y otros recursos para producir la página que ve el usuario en su pantalla.

A lo largo de esta explicación hemos mencionado en diversas ocasiones el término "HTML" como concepto fundamental para la creación de este invento. ¿Pero qué es realmente y cuál es el origen?

### 5.3 HTML

HTML es un lenguaje que se utiliza fundamentalmente en el desarrollo de páginas web.

HTML es la sigla de HyperText Markup Language (Lenguaje de Marcación de Hipertexto) es un lenguaje que se utiliza comúnmente para establecer la estructura y

contenido de un sitio web, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .htm o .html.

El lenguaje de HTML funciona por medio de "etiquetas" que describen la apariencia o función del texto enmarcado. Este lenguaje puede llegar a incluir un script o código que tenga incidencia en el comportamiento del navegador web de elección.



Figura 6. Ejemplo de Código HTML



Figura 5. Tim Berners-Lee

Como ya hemos comentado anteriormente, la historia de lenguaje de marcado de hipertexto está estrictamente vinculada al científico inglés Tim Berners-Lee, quien es considerado el padre de la World Wide Web (W3) gracias a la investigación en el tema, que lideró desde 1980, mientras trabajaba para la Organización Europea para la Investigación Nuclear (CERN, por sus siglas en inglés). El objetivo de Berners-Lee fue crear un

sistema hipertextual que permitiera conectar y compartir documentos entre diferentes computadores, sin importar el sistema operativo y aplicación Web (navegador).

El trabajo se apoyó en sistemas y protocolos de comunicación, los cuales habían sido desarrollados por organizaciones militares, académicas y científicas de países europeos y Estados Unidos. Además, TimBL, como también conocido Berners-Lee, recibió apoyo del ingeniero de sistemas Robert Cailliau (si Berners-Lee es el padre, Cailliau podría ser uno de los tíos de la WWW).

Para 1991, el equipo de trabajo del CERN presentó el primer documento, donde se hacía una descripción oficial del lenguaje de marcado Web. El título del texto fue: HTML Tags (etiquetas HTML). Este documento aún reposa en el archivo de la World Wide Web Consortium (W3C) y puede ser consultado por cualquier usuario.

En 1993, la IETF (Internet Engineering Task Force) realiza otro documento sobre el tema. Este sería el segundo intento por normalizar y estandarizar la producción de hiper-documentos basados en la propuesta de Berners-Lee.

El 22 de septiembre de 1995, luego de cuatro años de trabajo, el lenguaje es reconocido como estándar, gracias a la presentación de la versión 2.0.

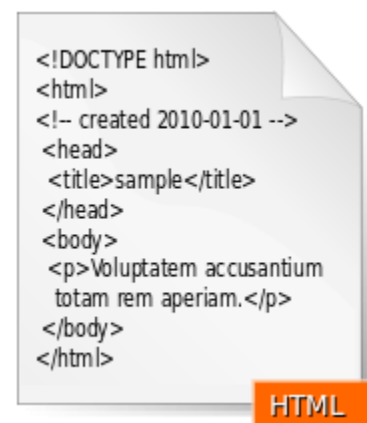


Figura 7. Documento HTML

## 5.4 Servidor

Nuestra página web necesitará de un dispositivo capaz de suministrarle información y todos los datos necesarios para llevar a cabo sus funcionalidades de forma correcta.

Es por eso que utilizaremos el microcontrolador LPC1768 como servidor. Un servidor es un equipo informático que forma parte de una red y provee servicios a otros equipos cliente. En nuestro caso contaríamos con un servidor de tipo compartido debido a que no solo suministra datos a un cliente, sino que también es capaz de generar por si solo esos datos y realizar operaciones con ellos.





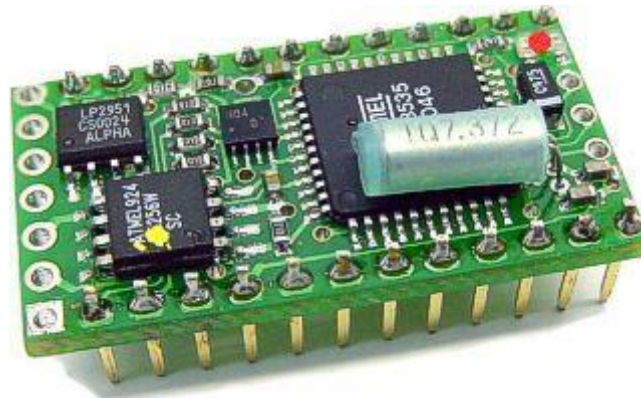
*Figura 8. Imagen de un numeroso grupo de servidores.*

### 5.5 Microcontrolador

Hemos mencionado varias veces la utilización del microcontrolador LPC1768 en este proyecto, pero todavía no hemos definido lo que es un microcontrolador.

Un microcontrolador (abreviado  $\mu\text{C}$ , UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades fundamentales de una computadora:

- Unidad de procesamiento
- Memoria
- Periféricos de entrada/salida



*Figura 9. Microcontrolador BasicX24*

El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario le escribe, es por esto que la programación es una actividad básica e indispensable cuando se diseñan circuitos y sistemas que los incluyan. El carácter programable de los microcontroladores simplifica el diseño de circuitos electrónicos. Permiten modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador.

Es por todas estas características por lo que hemos elegido un microcontrolador para llevar a cabo este proyecto.



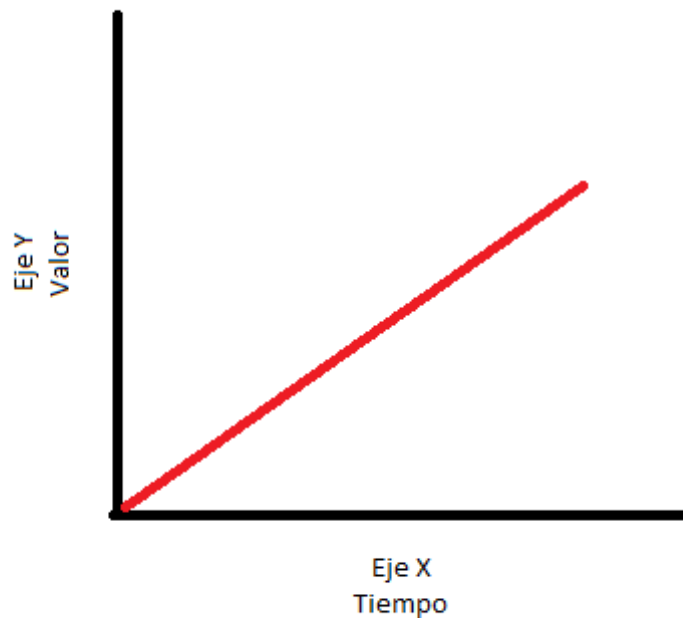
## 6. Objetivos y campo de aplicación

Este proyecto tiene como cometido final el de investigar la posibilidad de utilización del microcontrolador LPC1768 como servidor de una página web.

Para ello partimos del ejemplo provisto por easyweb, donde se nos presenta una sencilla aplicación mediante la cual se muestra de forma gráfica una serie de funcionalidades de manera automática.

A partir de este ejemplo, nuestra tarea es la de desarrollar un programa y una página web que se comuniquen entre sí y sean capaces de analizar datos y mostrarlos de forma gráfica.

En concreto nuestro proyecto tiene que ser capaz de medir valores de magnitudes reales durante un intervalo de tiempo predefinido y dibujar un gráfico con ellos donde su valor forme el eje Y y el tiempo sea el que forme el eje X.



*Figura 10. Ejemplo de gráfica mostrada*

En el ejemplo observamos cómo sería un gráfico de una magnitud que aumenta en función del tiempo.

Para llevar a cabo esta tarea vamos a valernos del periférico Analog to Digital Converter disponible en nuestro microcontrolador, por lo que seremos capaces de leer valores de tensión analógicas y dibujarlas.

Este proyecto es muy completo en el ámbito de la ingeniería ya que requiere tanto de conocimientos de informática como de electrónica.

Respecto al campo de la informática, para la realización de este proyecto es necesario disponer de nociones básicas de HTML, así como un conocimiento algo más avanzado del código C.

En cuanto al campo de la electrónica se refiere, es también útil haber trabajado con proyectos similares anteriormente ya que el conexionado de los cables y la gestión del microcontrolador requiere de ello.

## 7. Descripción del proyecto

El proyecto consta de dos partes claramente definidas y diferenciadas:

- La primera parte consiste en el análisis y la comprensión de distinto código de fabricante de la tarjeta Mini-DK2 que nos proporciona una serie de APIs para el desarrollo de proyectos sobre su microcontrolador. En esta parte comprobaremos el correcto funcionamiento de este código. Al terminar esta parte seremos capaces de asegurar el correcto funcionamiento de dicho código y seremos capaces de introducir modificaciones de cara a la segunda parte de nuestro proyecto.
- La segunda parte ocurre de forma consecutiva a la primera y consiste en el desarrollo e implementación de una serie de funciones que se basen en el ejemplo proporcionado y otorguen diversas funcionalidades a un servidor web creado sobre la tarjeta, contando con sus limitaciones.

En concreto, y como ya hemos perfilado en el apartado anterior, nuestro proyecto será capaz de interpretar magnitudes del mundo real (en nuestro caso voltaje), transformarlo en un valor digital y operar con ello.

Para realizar una aplicación más gráfica hemos incorporado un sensor de distancia, que será el que proporcione los voltajes que aparecerán en el gráfico. Esto provoca que se pueda observar como mediante la interposición de obstáculos, los valores de voltaje obtenidos en el microcontrolador, y por ende mostrados por la interfaz, varían en función de la distancia a estos obstáculos o la ausencia de los mismos.

De hecho, para incrementar este efecto, se ha implementado un algoritmo que traduce esos voltios captados en distancia. Este algoritmo viene descrito por el fabricante del sensor y en su apartado de requerimientos técnicos se explica de qué trata.

## 8. Requerimientos técnicos

Este proyecto ha requerido de la disposición de una serie de elementos hardware y software que se detallan a continuación

### 8.1 Requerimientos Hardware

#### **Microcontrolador**

El microcontrolador en cuestión se trata del LPC1768, un microcontrolador que ya hemos utilizado en otras asignaturas y que goza de numerosos periféricos, de entre los cuales destacamos:

- UART
- ADC
- DAC
- PWM
- TIMER
- EINT

Este microcontrolador se encuentra en la tarjeta Mini-DK2, la cual se basa en el cortexM3. Algunas de las características de esta tarjeta son:

- Hasta 512 kB de memoria flash.
- Hasta 64 kB de memoria de datos.
- Ethernet MAC.
- USB interfase.
- Un controlador de DMA de 8 canales.
- 4 UARTs.
- 2 canales CAN, 2 controladores SSP, y un SPI interfase.
- 3 I2C interfaces.
- I2S interfase.
- Un ADC de 12 bits y 8 canales de entrada y un DAC de 10-bit.
- 4 timers de propósito general.

- 6 salidas de PWM.
- Un reloj de tiempo real de bajo consumo con batería separada.
- Hasta 70 pines de entrada/salida de propósito general.

Es necesario remarcar que esta tarjeta consta de un conector estándar de tipo JTAG de 20 pines que será explicado más en detalle en el siguiente apartado.

Este micro se caracteriza por una adecuada velocidad de reloj de hasta 100MHz, así como un bajo consumo de energía. En adición a esto, dispone de todos los periféricos y requerimientos necesarios para la consecución del proyecto por un coste asumible y contaba con la ventaja de que ya habíamos trabajado con esta tarjeta con anterioridad por lo que no era necesario aprender la forma de programarla.

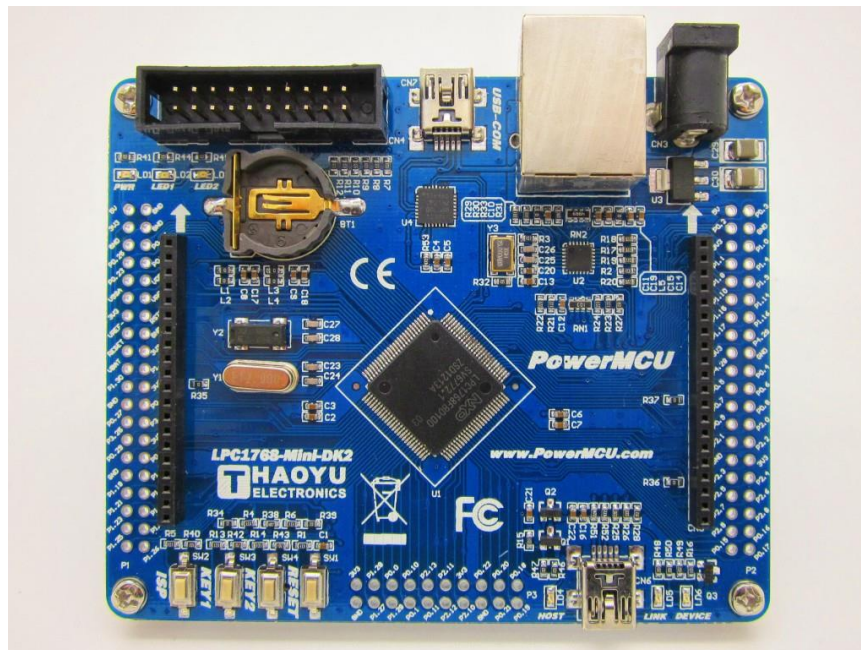


Figura 11. Tarjeta Mini-DK2 vista desde arriba

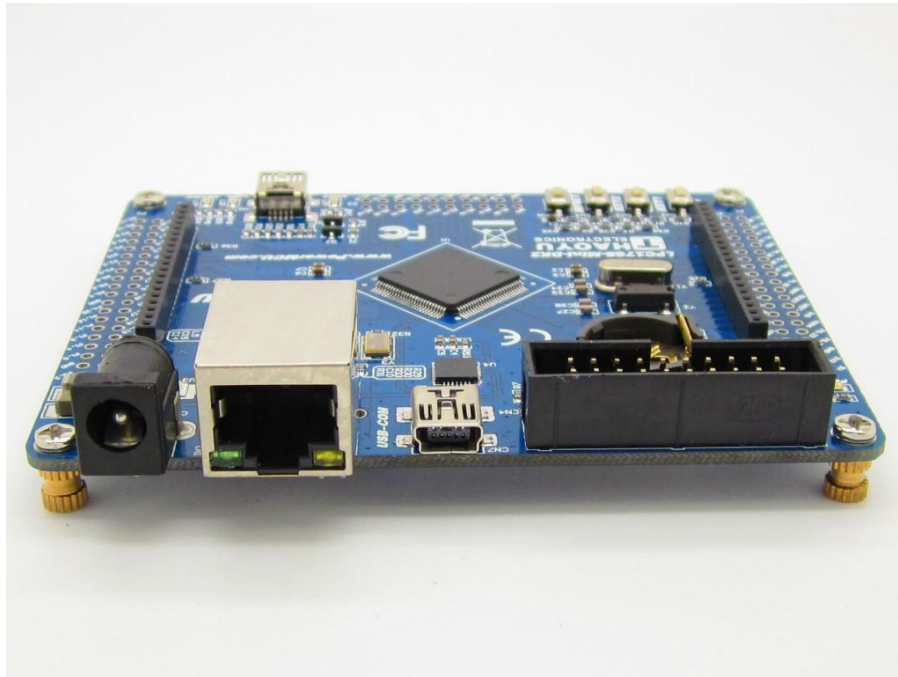
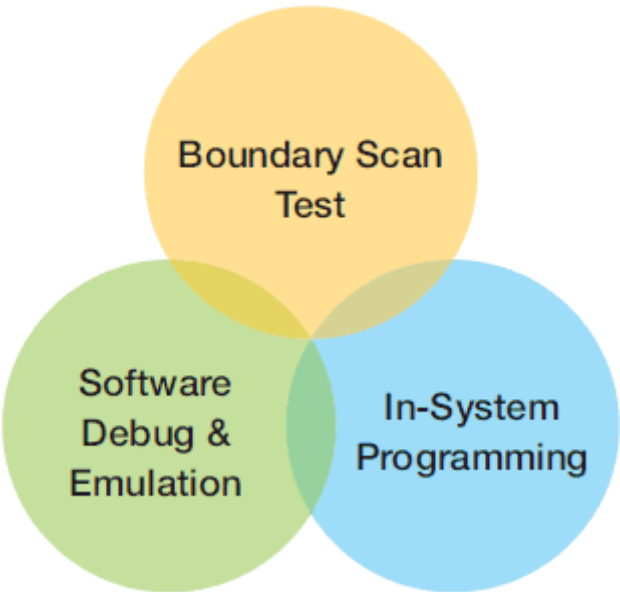


Figura 12. Alimentaciones de la tarjeta Mini-DK2. De izquierda a derecha: Alimentación de 5V, entrada Ethernet, entrada mini-USB y alimentación JTAG

### **Cable JTAG**

JTAG, es un acrónimo para Joint Test Action Group, es el nombre común utilizado para la norma IEEE 1149.1 titulada Standard Test Access Port (TAP) and Boundary-Scan Architecture para comprobar puertos de acceso utilizada para testear PCBs utilizando escaneo de límites.

Esta tecnología dispone de 4 cables y fue desarrollada para proveer de una tecnología



para el testeo de PCBA's (Printed Circuit Board Assemblies) sin necesidad del nivel de acceso físico requerido o la cantidad de desarrollo necesaria para un test funcional. El TAP fue diseñado para interactuar con nuevos registros que eran añadidos a dispositivos para implementar este método de comprobación.

Sin embargo, muy rápidamente se

Figura 13. Funcionalidades provistas por la tecnología JTAG

funcionalidades tales como la depuración y la programación.

El principal registro añadido a un dispositivo específicamente para JTAG es llamado Boundary Scan Register (BSR). Como su nombre indica, los bits de este registro se encuentran entre su núcleo principal y los pines a los que está conectada la placa.

Esta placa dispone de un conector ARM estándar de pines cuya descripción y disposición se muestra a continuación:

Signal	Connects to...	ARM 20-PIN Interface			
TMS	Test Mode State pin — Use 100K Ohm pull-up resistor to VCC.	VCC	1		2 VCC (optional)
TDO	Test Data Out pin.	TRST	3		4 GND
RTCK	JTAG Return Test Clock.	TDI	5		6 GND
TDI	Test Data In pin — Use 100K Ohm pull-up resistor to VCC.	TMS	7		8 GND
TRST	Test ReSeT/ pin — Use 100K Ohm pull-up resistor to VCC. TRST is optional and not available on some devices. You may leave it unconnected.	TCLK	9		10 GND
TCLK	Test CLock pin — Use 100K Ohm pull-down resistor to GND.	RTCK	11		12 GND
VCC	Positive Supply Voltage — Power supply for JTAG interface drivers.	TDO	13		14 GND
GND	Digital ground.	RESET	15		16 GND
RESET	RSTIN/ pin — Connect this pin to the (active low) reset input of the target CPU.	N/C	17		18 GND
		N/C	19		20 GND

Figura 14. Descripción de los pines JTAG de los que dispone la placa

Para llevar a cabo la conexión, hemos elegido el cable JTAG provisto por ULINK2 y que dispone de las siguientes características:

- Soporta varios dispositivos ARM7, ARM9, Cortex-M, 8051 y C166
- Velocidad de JTAG de hasta 10MHz
- Soporte de SWD (Serial Wire Debug) para dispositivos basados en el cortex-M
- Soporte de SWV (Serial Wire Viewer) de traza de datos para el Cortex-M de hasta 1Mbit/s (en modo UART)
- Memoria de escritura/lectura durante la ejecución, la emulación y la depuración
- Integrado en el software de Keil uVision
- Amplio rango de voltaje para el dispositivo objetivo: 2.7V-5.5V
- Alimentado por USB (no se requiere fuente de alimentación externa)
- Instalación Plug-and-Play (conectar y ejecutar) utilizando drivers estándar de Windows
- Conexión con 20 pines del conector estándar JTAG de ARM

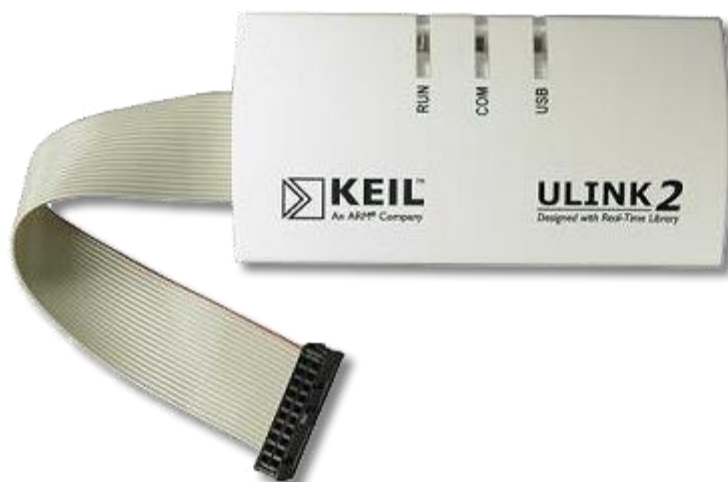


Figura 15. Cable ULINK2



### **Router WiFi**

Un router o enrutador es un producto de hardware que permite interconectar computadoras que funcionan en el marco de una red.



*Figura 16. Router inalámbrico*

El router se encarga de establecer qué ruta se destinará a cada paquete de datos dentro de una red informática.

Un router se vale de un protocolo de enrutamiento, que le permite comunicarse con otros enrutadores o encaminadores y compartir información entre sí para saber cuál es la ruta más rápida y adecuada para

enviar datos.

Un típico enrutador funciona en un plano de control (en este plano el aparato obtiene información acerca de la salida más efectiva para un paquete específico de datos) y en un plano de reenvío (en este plano el dispositivo se encarga de enviar el paquete de datos recibidos a otra interfaz).

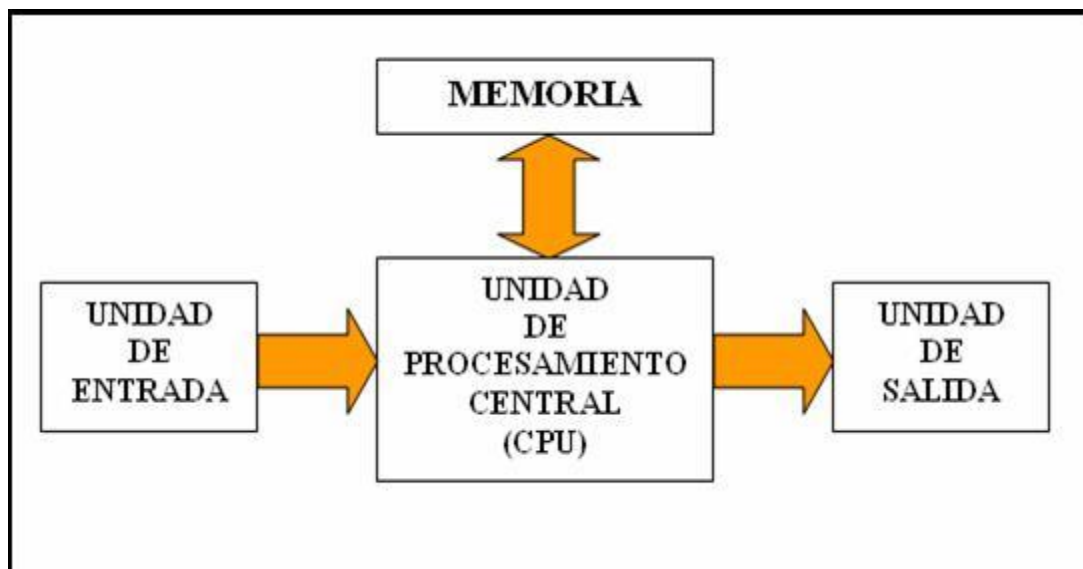
El router tiene múltiples usos más o menos complejos. En su uso más común, un enrutador permite que en una casa u oficina pequeña varias computadoras aprovechen la misma conexión a Internet. En este sentido, el router opera como receptor de la conexión de red para encargarse de distribuirlo a todos los equipos conectados al mismo. Así, se conecta una red o Internet con otra de área local.

En nuestro caso concreto, el router nos sirve tanto como para proveer de conexión al microcontrolador mediante el cable Ethernet, como para poder acceder a la página web donde se muestran los gráficos gracias a su conectividad inalámbrica.

### Ordenador

Un ordenador es una máquina electrónica que está formada, físicamente, por numerosos circuitos integrados y otros muchos componentes de apoyo, extensión y accesorios, que en conjunto pueden ejecutar tareas diversas con suma rapidez y bajo el control de un programa.

El ordenador, como ya sabemos, es una máquina que posee, al menos, una unidad central de procesamiento, una memoria principal y algún periférico o dispositivo de entrada y otro de salida. Los dispositivos de entrada permiten el ingreso de datos, la CPU se encarga de su procesamiento (operaciones aritmético-lógicas) y los dispositivos de salida los comunican a otros medios. Es así, que la computadora recibe datos, los procesa y emite la información resultante, la que luego puede ser interpretada, almacenada, transmitida a otra máquina o dispositivo o sencillamente impresa; todo ello a criterio de un operador o usuario y bajo el control de un programa.



*Figura 17. Esquema básico de la estructura de un ordenador*

El hecho de que sea programable, le posibilita realizar una gran diversidad de tareas, esto la convierte en una máquina de propósitos generales (a diferencia, por ejemplo, de una calculadora cuyo único propósito es calcular limitadamente). Es así que, sobre la base de datos de entrada, puede realizar operaciones y resolución de problemas en las más diversas áreas del quehacer humano (administrativas, científicas, de diseño, ingeniería, medicina, comunicaciones, música, etc), incluso muchas cuestiones que directamente no serían resolubles o posibles sin su intervención.

Básicamente, la capacidad de una computadora depende de sus componentes hardware, en tanto que la diversidad de tareas radica mayormente en el software que admita ejecutar y contenga instalado.

Este dispositivo es fundamental a la hora de llevar a cabo el proyecto debido a que el microcontrolador necesita de un programa para funcionar, y este es provisto (mediante la conexión JTAG ya explicada) por el ordenador.

### **Sensor de distancia**

Un sensor de distancia o transductor de distancia es un dispositivo que, una vez alimentado, genera una tensión en función de la distancia de obstáculos o señales al mismo.

Existen varios tipos de sensores de distancia:

- Interruptores de posición: También conocidos como finales de carrera, son dispositivos eléctricos, neumáticos o mecánicos situados al final del recorrido de un elemento móvil, como por ejemplo una cinta transportadora, con el objetivo de enviar señales que puedan modificar el estado de un circuito. Internamente pueden contener interruptores normalmente abiertos



Figura 18. Fin de carrera

(NA), cerrados (NC) o conmutadores dependiendo de la operación que cumplan al ser accionados.

- Capacitivos: La función del detector capacitivo consiste en señalar un cambio de estado, basado en la variación del estímulo de un campo eléctrico. Los sensores capacitivos detectan objetos metálicos, o no metálicos, midiendo el cambio en la capacitancia, la cual depende de la constante dieléctrica del material a detectar, su masa, tamaño, y distancia hasta la superficie sensible del detect



*Figura 19. Sensor de distancia capacitivo*

están contruidos sobre la base de un oscilador LC. Debido a la influencia del objeto a detectar, y del cambio de capacitancia, la amplificación se incrementa haciendo entrar en oscilación el oscilador.

- Inductivos: Los sensores inductivos de proximidad han sido diseñados para trabajar generando un campo magnético y detectando las pérdidas de corriente de dicho campo generadas al introducirse en él los objetos de detección férricos y no férricos.

El sensor consiste en una bobina con núcleo de ferrita, un oscilador, un sensor de nivel de disparo de la señal y un circuito de salida.



Figura 20. Sensor de distancia inductivo con salida analógica

- Fotoeléctricos: El receptor de rayos infrarrojos suele ser un fototransistor o un fotodiodo. El circuito de salida utiliza la señal del receptor para amplificarla y adaptarla a una salida que el sistema pueda entender. La señal enviada por el emisor puede ser codificada para distinguirla de otra y así identificar varios sensores a la vez



Figura 21. Sensor fotoeléctrico de reflexión directa

- Ultrasónico: Los sensores de ultrasonidos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias de hasta 8m. El sensor emite impulsos ultrasónicos. Estos reflejan en un objeto, el sensor

recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración. Estos sensores trabajan solamente en el aire, y pueden detectar objetos con diferentes formas, superficies y de diferentes materiales. Los materiales pueden ser sólidos, líquidos o polvorientos, sin embargo, han de ser deflectores de sonido. Los sensores trabajan según el tiempo de transcurso del eco, es decir, de emisión y el impulso del eco.



Figura 22. Sensor de distancia ultrasónico modelo hc-sr04

- Magnético: Los sensores de proximidad magnéticos son caracterizados por la posibilidad de distancias grandes de la conmutación, disponible de los sensores con dimensiones pequeñas. Detectan los objetos magnéticos (imanes generalmente permanentes) que se utilizan para accionar el proceso de la conmutación. Los campos magnéticos pueden pasar a través de muchos materiales no magnéticos, el proceso de la conmutación se puede también accionar sin la necesidad de la exposición directa al objeto.



*Figura 23. Sensor de proximidad magnético*

Como ya hemos explicado en apartados anteriores, la utilidad de un sensor de distancia en nuestro proyecto es la de obtener una fuente visual de medidas de tensión.

De este modo conseguimos que se aprecie en el gráfico mostrado por la interfaz web, alteraciones en la distancia de los obstáculos hasta el sensor durante la ejecución del programa.

Teniendo en cuenta las características deseadas, escogimos el sensor infrarrojo GP2D12 2Y0A21 de la compañía Sharp, debido a que ofrecía una implementación sencilla y eficaz, así como unos resultados suficientemente precisos y fiables para la aplicación que va a desarrollar.

El componente electroóptico GP2D12 está basado en un LED de infrarrojos y un optodetector PSD que proporciona una tensión de salida en función de la posición en la que incide el haz de luz que reciba. Su uso como medidor de distancia se basa en utilizar el LED para emitir un haz de luz que rebotará en obstáculo (en caso de que exista) y determinar la distancia a éste mediante el ángulo de incidencia del haz de luz generado por el LED y reflejado por el obstáculo sobre el PSD

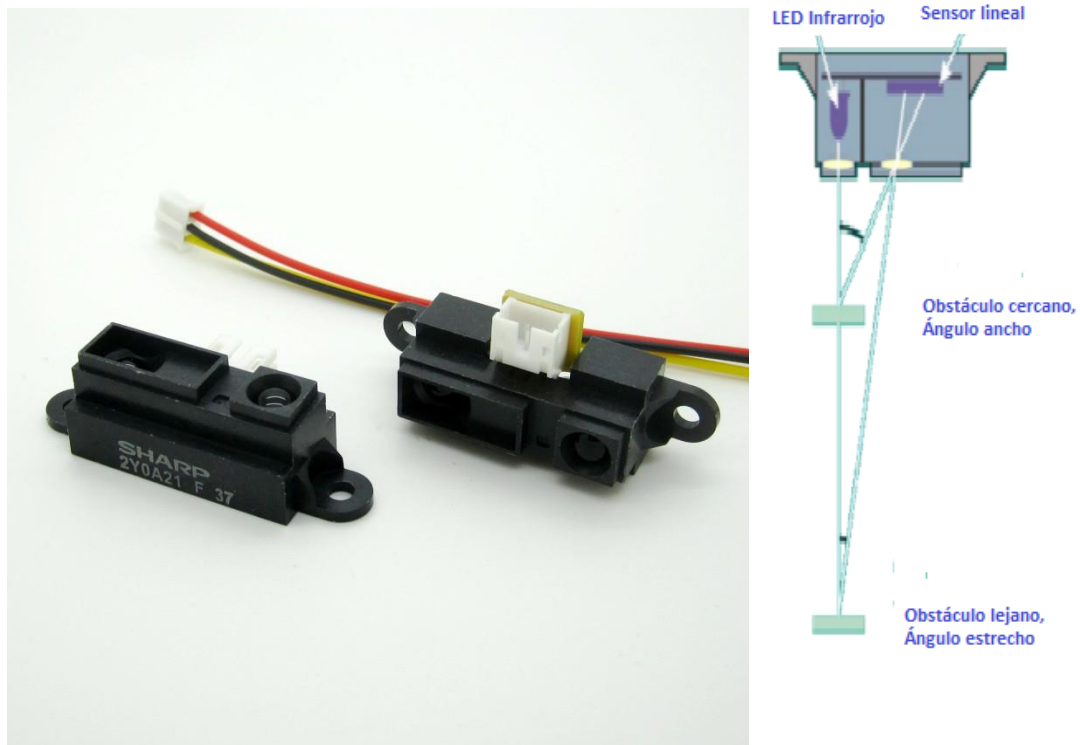


Figura 24. Imagen del sensor infrarrojo GP2D12

Estas imágenes detallan las características físicas y de conexionado interno del sensor:



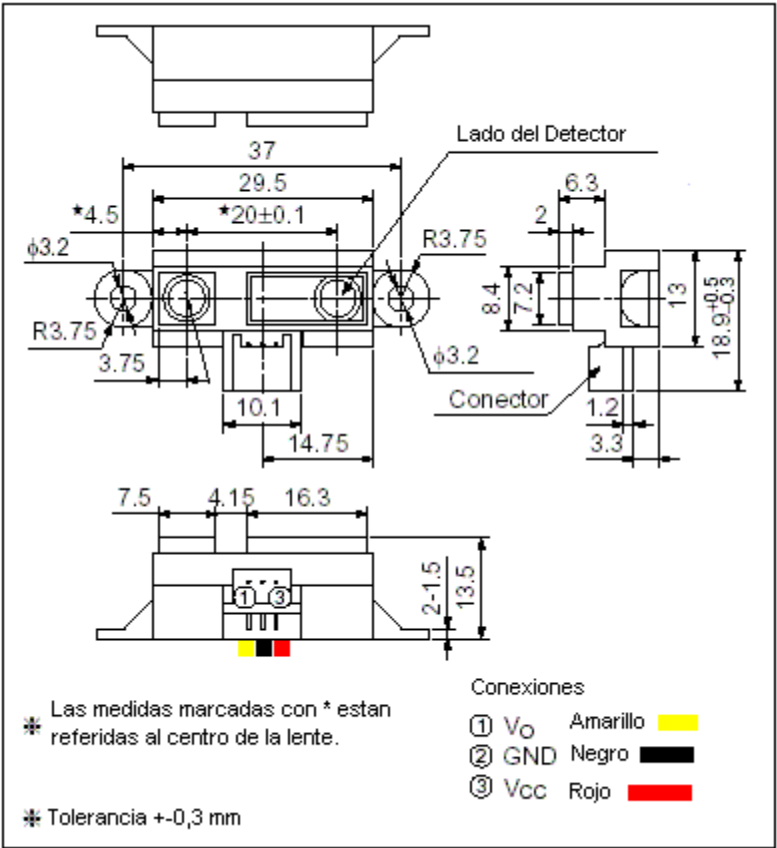


Figura 25. Dimensiones del sensor

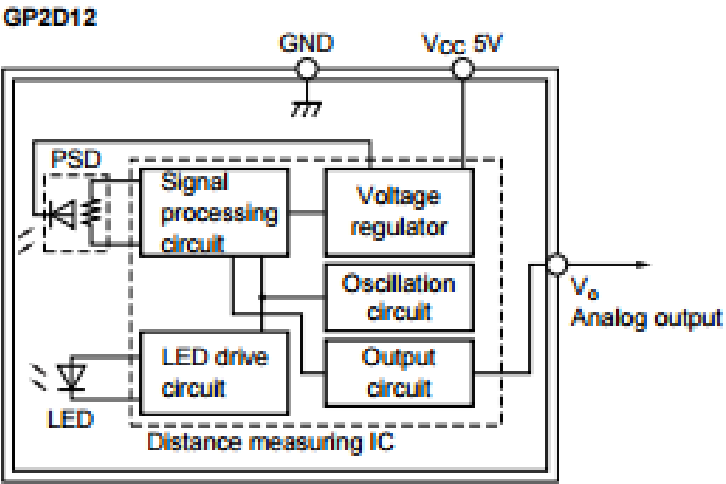
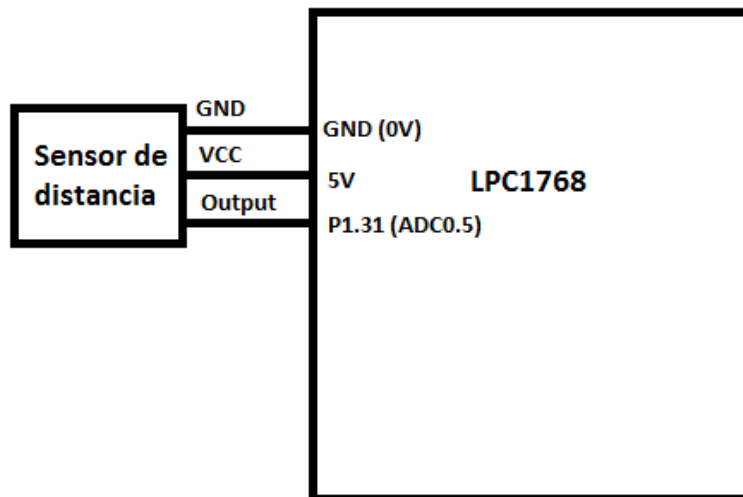


Figura 26. Conexionado de bloques del sensor.

En la figura 24 observamos como el sensor dispone de 3 cables de distintos colores: rojo, amarillo y negro.

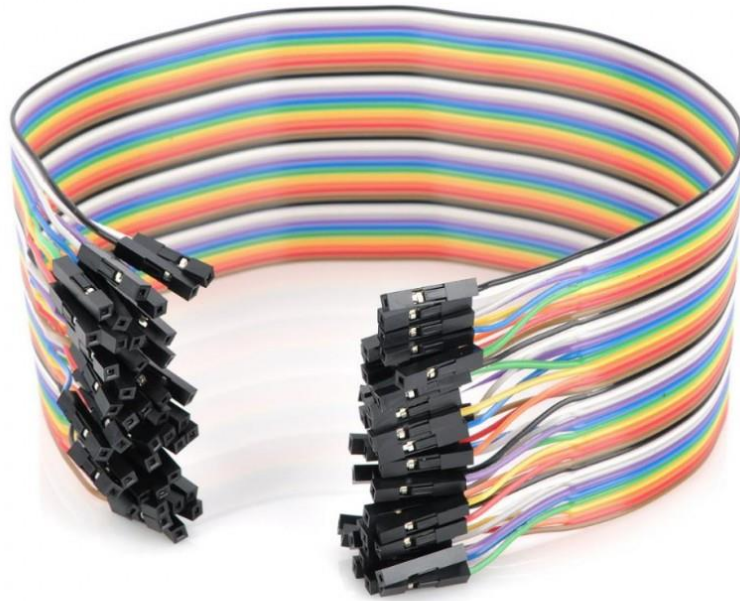
En la figura 25 se muestra que el cable amarillo equivale a la tensión de salida (Output), el cable negro equivale a masa (GND) y el rojo a la tensión de alimentación (VCC).



*Figura 27. Conexión entre el sensor y el microcontrolador.*

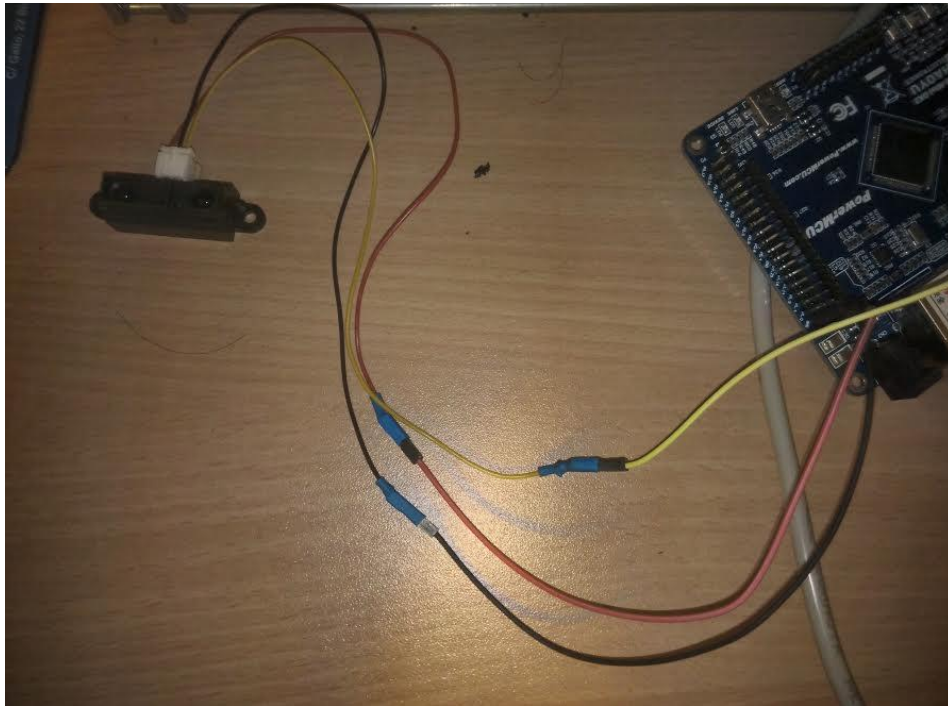
En esta imagen se muestra como se realiza la conexión entre el sensor de distancia y la tarjeta Mini-DK2, conectando los dos cables de alimentación a sendos pines de la tarjeta, mientras que la salida de tensión analógica del sensor se conecta al canal 5 del periférico ADC del que dispone la tarjeta. Es reseñable el hecho de que se ha elegido el canal 5 del periférico debido a que los canales 0 y 1 (también disponibles ya que no están ocupados con ninguna otra funcionalidad), no funcionaban de forma correcta.

Es también necesario señalar que los cables salientes del sensor eran cables normales, mientras que lo necesario para realizar la conexión con la placa eran cables finalizados en pines-hembra.



*Figura 28. Cables hembra-hembra*

Debido a esto, adquirimos un cable termoeléctrico para, mediante la aplicación de calor, unir uno de los cables mostrados en la figura superior con los cables provistos por el sensor, siendo éste el resultado.



*Figura 29. Imagen de la conexión mediante cables termoeléctricos*

Gracias a la utilización de estos cables logramos que la unión fuese más segura y no hubiera desconexiones de forma esporádica.

Otro factor muy a tener en cuenta a la hora de utilizar este sensor, es el hecho de que éste no proporciona una salida lineal en función de la distancia, sino que esta es la respuesta:

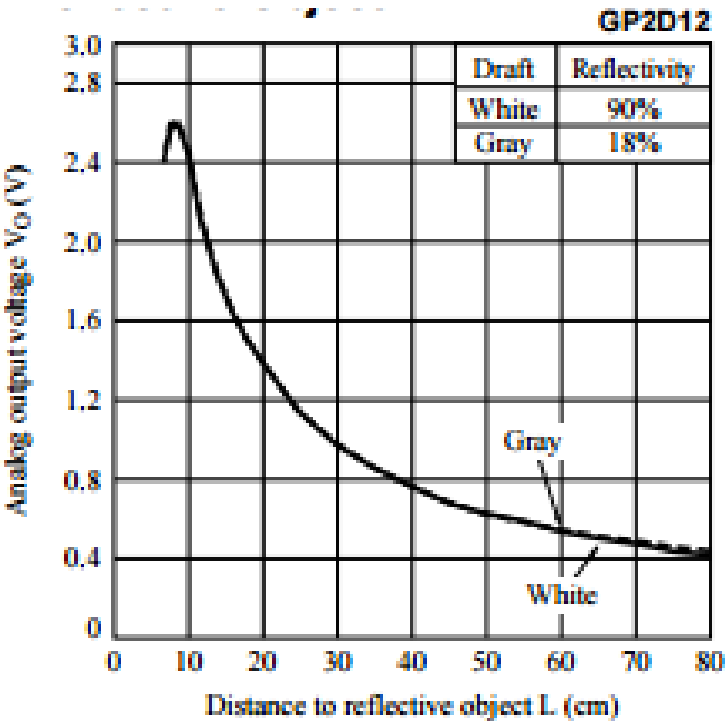


Figura 30. Respuesta de la tensión en función de la distancia proporcionada por nuestro sensor.

Debido a que necesitamos linealizar la función lo máximo posible, invertimos la gráfica obteniendo esta función.



Figura 31. Función del sensor invertida.

Como vemos, esta función es mucho más lineal en el rango de 10-80cm que es el rango que vamos a utilizar en nuestra aplicación.

Utilizando el segundo de los métodos indicados por el fabricante, se puede obtener la función lineal aproximada que relaciona la tensión de salida del GP2D12  $V_{out}$  con la distancia  $L$  [cm] a la que se encuentra el objeto de reflexión (el obstáculo, en este caso) de este modo:

$$V_{out} = \frac{2.1 - 0.2}{0.08} \left( \frac{1}{L + 0.42} \right) + 0.2$$

Despejando  $L$ ,

$$L = \frac{23.75}{V - 0.2} - 0.42 \text{ (cm)}$$

### Cable Ethernet

El microcontrolador necesita estar conectado a Internet para poder funcionar como servidor de la página web. Esta conexión se consigue mediante un cable que conecta el enrutador con la entrada Ethernet de la placa.



Figura 32. Imagen de cable Ethernet

Hemos utilizado concretamente un cable de categoría 5E que logra alcanzar unas velocidades máximas de hasta 1000MB/s, velocidad más que suficiente para la funcionalidad deseada.

	Distancia	Velocidad Máxima (Mb/s)				PoE	Mhz
		10	100	1.000	10.000		
Cat-5	100	X	X			X	100
Cat-5e	100	X	X	X		X	100
Cat-6	100	X	X	X		X	250
Cat-6a	100	X	X	X	X	X	500

Figura 33. Tabla comparativa entre distintos cables Ethernet

### **Cable USB-MiniUSB**

El Bus Universal en Serie (BUS) (en inglés: Universal Serial Bus), más conocido por la sigla USB, es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.

Este bus es el encargado de alimentar la tarjeta, lo cual realiza mediante una fuente de corriente continua, generalmente a 5V. En nuestro proyecto esta fuente de energía puede ser tanto un puerto del ordenador, como una batería extraíble que se podría incorporar en caso de ser necesario transportar el montaje.

## 8.2 Requerimientos Software

Además de todos los dispositivos hardware comentados, para la realización del proyecto han sido necesarios un conjunto de elementos software que se explican a continuación.

### **Keil uVision**

Keil es un entorno de desarrollo integrado (IDE) que combina la gestión de proyectos,



*Figura 34. Portada del entorno de desarrollo Keil*

entorno de ejecución, facilidades de construcción, edición de código fuente, y depuración del programa en un único entorno. uVision soporta la capacidad de utilizar múltiples pantallas y permite la creación de ventanas individuales en cualquier parte

de la interfaz.

El depurador provee un entorno en el que se puede comprobar, verificar y optimizar el código de tu aplicación. Incluye características tradicionales como los puntos de parada,



las ventanas de observación, o el control de la ejecución y aporta plena visibilidad de los periféricos del dispositivo. Este entorno se puede dividir en varias partes:

- Entorno de ejecución y gestión de proyectos: Gracias a este entorno se pueden crear aplicaciones software utilizando componentes y dispositivos predefinidos provistos por los Paquetes Software. Estos componentes contienen librerías, módulos fuente, archivos de configuración, plantillas de código fuente y documentación. Estos componentes son genéricos en un amplio rango de dispositivos y aplicaciones.
- Editor: El editor proporcionado por Keil incluye todas las características estándar de un editor de código moderno y también está accesible durante la depuración. La sintaxis de código, resalto de texto y demás funcionalidades se encuentran optimizadas para C y C++.

Este software permite además simular un programa sin un dispositivo Hardware real, mediante el modo simulación, a la vez que proporciona archivos .hex y .axf

Todo esto unido al hecho de que es un software gratis y que ya hemos trabajado con anterioridad en algunas asignaturas lo convierte al candidato perfecto para este proyecto.

Utilizamos concretamente la versión 4.72a de este software

### **Drivers**

Los drivers necesarios para este proyecto fueron los que comunicaban el ordenador con nuestro cable ULINK2. Estos drivers debían instalarse de forma automática al conectar el cable por USB, pero no fue este el caso por lo que hubo que entrar en la página de Keil y descargarlos manualmente.

Una vez seguido este procedimiento el cable comenzó a funcionar como esperaba (antes el programa Keil no lo reconocía) y no hubo más problemas en este aspecto.

## JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque



*Figura 35. Logo de Javascript*

existe una forma de JavaScript del lado del servidor. Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

- Client-side: JavaScript extiende el núcleo del lenguaje proporcionando objetos para controlar un navegador y su modelo de objetos (o DOM, por las iniciales de Document Object Model). Por ejemplo, las extensiones del lado del cliente permiten que una aplicación coloque elementos en un formulario HTML y responda a eventos del usuario, tales como clicks del ratón, ingreso de datos al formulario y navegación de páginas.
- Server-side: JavaScript extiende el núcleo del lenguaje proporcionando objetos relevantes a la ejecución de JavaScript en un servidor. Por ejemplo, las extensiones del lado del servidor permiten que una aplicación se comunique con una base de datos, proporcionar continuidad de la información de una invocación de la aplicación a otra, o efectuar manipulación de archivos en un servidor.

En nuestro caso hemos utilizado un JavaScript de tipo del lado del cliente, ya que la utilidad que queremos otorgarle es la de dibujar gráficos según los datos que hayamos escrito en su código HTML.

Es común confundir JavaScript con Java pero tienen unas cualidades fundamentales que los diferencian. El lenguaje JavaScript se parece a Java pero no tiene el tipo estático (static) de Java, ni tiene un chequeo de tipos fuerte. JavaScript usa la mayoría de la sintaxis de expresiones de Java, convenciones de nombrado, y las construcciones básicas de control de flujo, razón por la cual se le cambió el nombre del original LiveScript a JavaScript.

En contraste con el sistema de clases construidas por declaraciones que se usa en tiempo de compilación de Java, JavaScript soporta un sistema de tiempo de ejecución basado en un pequeño número de tipos de datos que representan valores numéricos, lógicos, y de cadena de caracteres (string). JavaScript tiene un modelo de objetos basado en prototipos en lugar del modelo de objetos basado en clases, que es más común. El modelo basado en prototipo proporciona herencia dinámica; esto es, que lo que se hereda puede variar entre objetos individuales. JavaScript también soporta funciones sin ningún requerimiento declarativo especial. Las funciones pueden ser propiedades de los objetos, ejecutándose como métodos levemente tipados.

Comparado con Java, JavaScript es un lenguaje muy libre de forma. No hay que declarar todas las variables, clases, y métodos. No hay que preocuparse de si los métodos son públicos, privados, o protegidos, y no hay que implementar interfaces. Las variables, parámetros, y retornos de funciones no tienen que declararse explícitamente de un tipo dado.

Java es un lenguaje de programación basado en clases, diseñado para lograr seguridad de tipos, y ejecución rápida. Seguridad de tipos significa, por ejemplo, que no es posible hacer una conversión de tipos a un Integer de Java para obtener una referencia a un objeto, o acceder a memoria protegida mediante la alteración del bytecode de una clase. El modelo basado en clases de Java implica que los programas consisten exclusivamente en clases y sus métodos. La herencia de clases de Java y el tipado fuerte generalmente requiere que jerarquías de objetos fuertemente acopladas. Estos requerimientos hacen que la programación en Java sea más compleja que la programación en JavaScript.

En contraposición, JavaScript proviene en espíritu de una línea de lenguajes de programación más pequeños, con tipado dinámico, como HyperTalk, y dBASE. Estos lenguajes de scripting hacen accesibles las herramientas de programación a audiencias mucho más amplias. Esto se debe a su sintaxis más indulgente, funcionalidad especializada ya incluida, y mínimos requisitos para la creación de objetos.

Como ya especificaremos más adelante, hemos optado por el javascript propuesto por google debido a que se trata de un software gratuito y muy sencillo de implementar. Simplemente necesitamos descargar un paquete de librerías y elegir algunos parámetros de configuración para disponer de gráficos muy vistosos y elegantes.

Además, permite la elección entre una innumerable cantidad de gráficos distintos. En nuestro caso hemos elegido el gráfico de línea, ya que pensamos que representa los datos de una

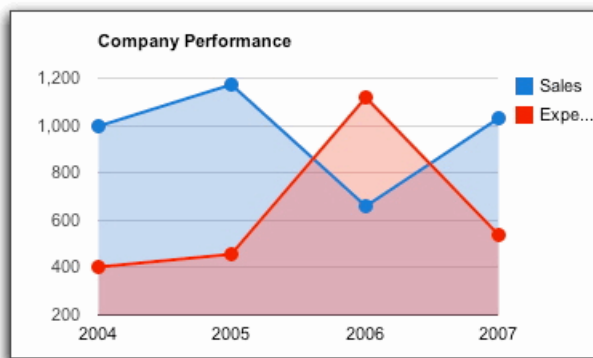


Figura 37. Ejemplo de gráfico de línea

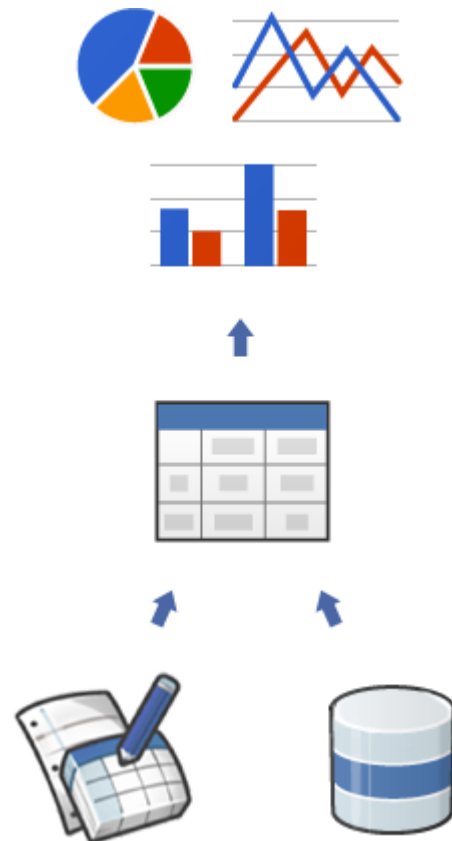


Figura 36. Imagen esquemática de Google Charts

## 9. Esquema del conexionado

Una vez expuestos todos los elementos utilizados, procedemos a mostrar un esquema de cómo queda la conexión de todos los elementos entre sí.

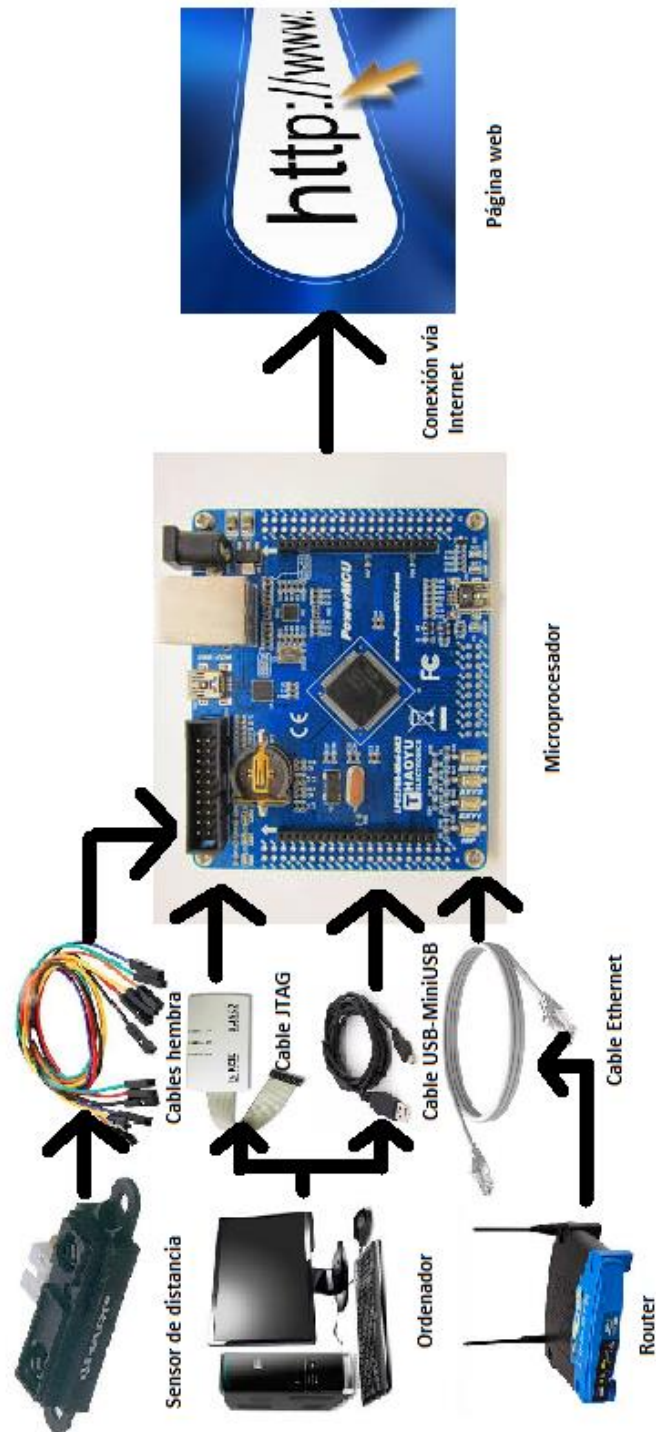


Figura 38. Esquema de conexionado

En la imagen disponemos de un resumen visual del conexionado de todos los elementos:

- Comenzamos por el sensor de distancia, que recibe valores de tensión analógicos en función de la distancia hasta el obstáculo (siempre que esté alimentado), y se lo comunica al microcontrolador a través de los cables hembra.
- El ordenador se encarga de alimentar a 5V el microcontrolador mediante el cable USB-MiniUSB, y también carga el programa obtenido gracias al entorno de desarrollo de Keil uVision mediante el cable JTAG.
- El router otorga la conectividad a Internet necesaria para que el microcontrolador sea capaz de comunicarse con la página web.
- Una vez que el micro dispone de todos los datos necesarios, realiza los algoritmos detallados anteriormente y se los envía a la página web, que los muestra de forma gráfica.

## 10. Memoria

El primer paso para la consecución de este proyecto era el de analizar y entender el funcionamiento del programa de ejemplo proporcionado por easyweb que nos serviría de base para realizar las funcionalidades que queríamos implementar. Una vez comprendido este funcionamiento, comenzaríamos a desarrollar el código necesario y las pruebas pertinentes, hasta conseguir lograr el objetivo del proyecto.

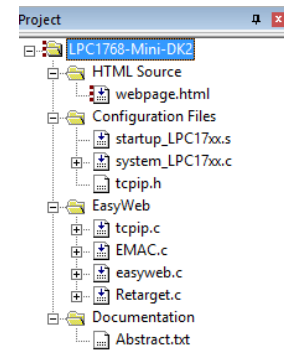


Figura 39. Esquema de los archivos que conforman el proyecto.

### 10.1 Análisis del programa

Observamos que el programa está compuesto de diversos archivos “.c”, de los cuales se derivan otros archivos y bibliotecas necesarios para la comunicación entre la página web y el microcontrolador.

De entre los archivos, nos dimos cuenta que el encargado de añadir funcionalidades y gestionarlas era el llamado “easyweb.c”, mientras que los otros gestionaban todo el aspecto de la comunicación entre el micro y la página web, y por ende empezamos a analizarlo. Este código se encuentra disponible en el Anexo 1 de este mismo proyecto.

El programa comienza en el main, desde donde se configura el SysTick, el ADC0.5 (P1.31) y el LED P2.10 como salida. El systick interrumpe cada 10ms, y a las 20 interrupciones (200ms) conmuta el LED.

Después llama a la función TCPLowLevelInit que inicializa algunos valores necesarios para la comunicación.

Con un while (1), espera a la llegada de instrucciones, y una vez llegan, llama a las funciones DoNetworkStuff y HTTPServer.

```

TCPLowLevelInit();
while (1)
{
    if (!(SocketStatus & SOCK_ACTIVE)) TCPPassiveOpen();
    DoNetworkStuff();

    HTTPServer();
}

```

Figura 40. Código easyweb situado en la función main

DoNetworkStuff es una función que, como su nombre indica, realiza una serie de operaciones cuando se establece la conexión que permiten la comunicación entre el microprocesador y el servidor.

La función HTTPServer es donde se produce todo el intercambio de información que nos interesa en este proyecto entre el código C y el código html y es por ello que la analizamos más profundamente. Esta función consiste en comprobar si es posible realizar la comunicación, y comienza a transmitir el código html – que está almacenado en un array de caracteres llamado “Webside” al que apunta el puntero “PWebSide” – en bloques de 512 bits (valor registrado en la constante llamada MAX\_TCP\_TX\_DATA\_SIZE), y cuando ya queda menos de esta cantidad, transmite los bits restantes.

La primera transmisión de datos transmite el encabezado, que se encuentra almacenado en el string “GetResponse”, y luego mueve el puntero hacia adelante para que en el resto de los 512 bits se empiece a transmitir código html.

```

const unsigned char GetResponse[] =
{
    "HTTP/1.0 200 OK\r\n"
    "Content-Type: text/html\r\n"
    "\r\n"
};

```

Figura 41. Definición de la cadena de caracteres GetResponse

En la última transmisión, almacena en TCP\_TX\_BUF (que se trata del puntero que apunta al código html y por lo cual, donde se escriben el código que se desea transmitir), los últimos bits a transmitir, y cierra la conexión mediante la función TCPClose.



Independientemente del número de transmisiones realizadas, para realizar estas transmisiones, una vez que los punteros ya apuntan adonde se desea, se llama a las funciones InsertDynamicValues y TCPTransmitTxBuffer.

- El valor del ADC se escribe en la función referenciada anteriormente InsertDymnamicValues. En esta función se apunta el puntero Key hacia el vector TCP\_TX\_BUF. Posteriormente, se recorre todo el vector TCP\_TX\_BUF (que a su vez apunta al código html), buscando la combinación de letras ADx% y hace tres tareas en función del valor de x:

- si x es 8 (AD8%) almacena un valor aleatorio dividido por 4024 en la variable adcValue

- si x es 7 (AD7%) multiplica este valor por 100 y divide por 4024 y lo almacena en newKey mediante un sprintf

- si x es 1 (AD1%) aumenta el contador de la página.

```

if (*Key == 'A')
if (*(Key + 1) == 'D')
if (*(Key + 3) == '\0')
switch (*(Key + 2))
{
case '8' : // "AD8%"?
{
adcValue = rand()%4024; // get AD value
sprintf(NewKey, "0x%03X", adcValue); // insert AD converter value
memcpy(Key, NewKey, 5);
break;
}
case '7' : // "AD7%"?
{
sprintf(NewKey, "%3u", (adcValue*100)/4024); // copy saved value from previous read
memcpy(Key, NewKey, 3);
break;
}
case '1' : // "AD1%"?
{
sprintf(NewKey, "%3u", ++pagecounter); // increment and insert page counter
memcpy(Key, NewKey, 3);
*(Key + 3) = '\0';
break;
}
}
}

```

Figura 42. Escritura en el puntero Key el valor que se desea escribir en el código html

En el código html vienen incluidas estas tres opciones. La primera es para mostrar el valor del ADC, la segunda para crear una barra de la anchura del ADC (es decir, si el ADC

valiera la mitad del valor máximo la barra mediría la mitad del valor máximo), y la tercera sirve para contar el número de conversiones.

Tras hacer esto, realiza un memcpy (string.h), que lo que hace es copiar el valor del segundo miembro (en este caso newKey) en el primer miembro (Key).

Este newKey, es un buffer cuyo valor se ha almacenado mediante un sprintf (stdio.h), del valor que habíamos escrito previamente en adcValue o cualquier otra variable. El formato "0x%03X" implica que lo que se almacena es el valor, precedido de "0x", convertido a hexadecimal (X) y con un ancho mínimo de 3 caracteres (03). El formato "%3u" implica que el tamaño mínimo es de 3 caracteres (3) y que es un entero sin signo (u).

Tras haber almacenado el valor deseado en Key (que lo que hace es almacenarlo en TCP\_TX\_BUF y, por ende, escribirlo en el propio código html), se aumenta su valor para que apunte al siguiente componente del vector TCP\_TX\_BUF.

## 10.2 Desarrollo de funcionalidades

Una vez que hemos estudiado el funcionamiento del programa, pasamos a comprobarlo en la placa para ver que el programa funciona correctamente. Para la correcta simulación, primero compilamos y descargamos el programa en nuestro

microprocesador LPC1768, lo que realizamos mediante un cable JTAG, concretamente el ULINK 2.



*Figura 43. Imagen del dispositivo JTAG. Este es el modelo ULINK 2, empleado para la realización de este proyecto.*

Es necesario añadir que antes de ejecutar el programa era necesario realizar algunas modificaciones para simular el programa de forma que se adaptase a las condiciones de nuestro internet. Para ello, entramos en el archivo “tcpip.h” que consiste en una biblioteca que almacena valores relacionados con los proporcionados por la conexión del usuario. Modificamos los parámetros de Dirección IP, Máscara de Subred y Puerta de enlace con los valores que nos proporciona una búsqueda en las propiedades de nuestra conexión. Es necesario resaltar que, en el parámetro de la Dirección IP, hay que elegir un valor que concuerde en los tres primeros números con el resto de aparatos conectados al router Wifi, pero darle un cuarto valor que se encuentre sin utilizar de forma que no interfiera con cualquier otro aparato.

Destacar que, por motivos desconocidos, las primeras simulaciones no tuvieron los resultados deseados (no se produjo conexión entre la página web y el microprocesador), pero estos problemas se arreglaron gracias a la utilización de otra placa distinta, por lo

que suponemos que la primera utilizada tenía algún tipo de avería que ocasionaba fallos al intentar realizar esta simulación en concreto.

Una vez incorporada la nueva placa, observamos como el programa se desarrolla de forma que la página web se actualiza cada 0.4 segundos, aumentando en 1 el contador que se encuentra abajo en color gris, a la vez que muestra un valor que, de momento, se genera de forma aleatoria y lo pone en una tabla con fondo rojo.

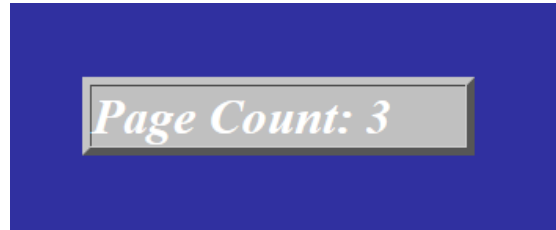


Figura 44. Imagen del contador de la página que se incrementa en 1 en cada actualización de la misma.

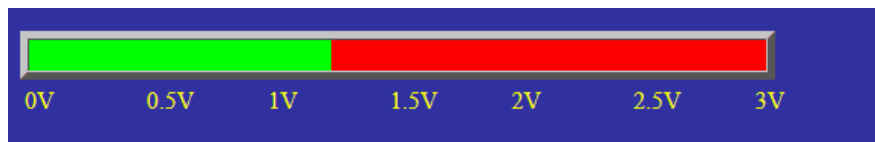


Figura 45. Imagen de la barra con fondo rojo donde el apartado verde indica el valor que se está midiendo.

Ahora que hemos visto los resultados de la simulación, terminamos de definir los objetivos del proyecto:

- Mientras que el programa original solo trabaja con un dato a la vez, nuestro proyecto será capaz de interpretar y mostrar varios datos por cada actualización de la página.
- Nuestro programa utilizará el periférico ADC (Analog to Digital Converter) para ver desde la interfaz web variables observables en el mundo real.
- Nuestro programa estará planteado de forma que resulte flexible ante posibles futuras modificaciones.

Dado que los objetivos ya están definidos, llega el momento de plantear el procedimiento a seguir para lograr la funcionalidad deseada.

Para tener la capacidad de mostrar gráficos de forma sencilla en la web, hemos utilizado un programa javascript, que ya hemos comentado anteriormente en que consiste.

Primero comenzamos utilizando el javascript Canvas ([www.canvasjs.com](http://www.canvasjs.com)) pero más tarde cambiamos al Google Chart javascript (<https://developers.google.com/chart/>), debido a su mayor sencillez y facilidad en su uso.

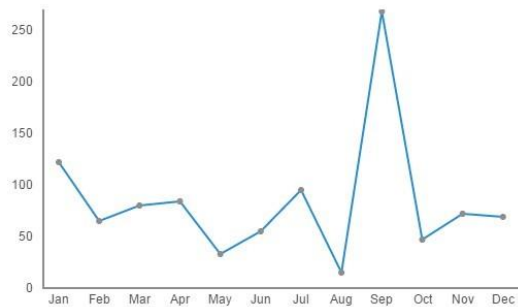


Figura 47. Imagen de gráfico creado mediante el javascript de Canvas

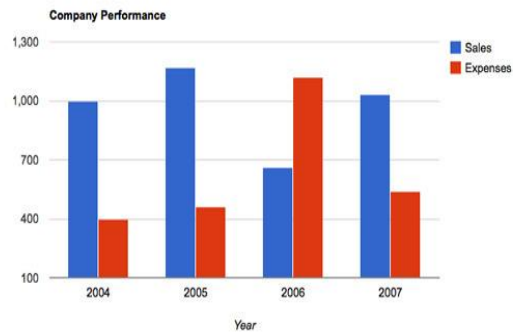


Figura 46. Imagen de gráfico creado mediante el javascript de Google

En adición a lo comentado anteriormente, el javascript proporcionado por Google permitía innumerables opciones de personalización y diseño de los gráficos, algunas de las cuales han sido añadidas al proyecto tal y como se indicará más adelante.

Una vez descargado este javascript, realizamos algunas pruebas en un archivo .html cualquiera para comprobar el correcto funcionamiento del programa y empezar a perfilar como iba a ser el gráfico que íbamos a mostrar y otros parámetros tales como títulos, colores etc.

Basándonos en el código original, llamamos a los datos por el nombre de D1%, D2%, D3% y D4% (en el código original se llamaban ADx), para que cuando desde el código C se lea el código html, sepamos que es ahí donde se deben introducir los valores adquiridos mediante el ADC.

```
var data = google.visualization.arrayToDataTable([  
  ['Texto', 'Valor numérico'],  
  ['Valor1', D1%],  
  ['Valor2', D2%],  
  ['Valor3', D3%],  
  ['Valor4', D4%]  
]);
```

Figura 48. Imagen que muestra extracto del código html con los valores que se muestran en el gráfico

En la imagen observamos como en el lugar donde se debería situar el apartado numérico del valor que deseamos dibujar en el gráfico, escribimos la combinación de caracteres Dx%, para que más adelante durante la lectura de este código escribamos en estos puntos los valores obtenidos durante la ejecución del programa. De esta forma es como se consigue que los gráficos varíen de forma sin manipular el código.

Viendo que el programa se comportaba conforme a lo esperado, pasamos a desarrollar algunas partes del código necesario para implementar las modificaciones descritas. Comenzamos configurando el ADC para que interrumpiese mediante el SysTick. Esto lo planteamos de forma que cuando el SysTick llegara a un número de interrupciones determinado (cabe recordar que en el código original el SysTick interrumpía cada 10 milisegundos) propiciara la conversión inmediata del ADC.

También programamos la función de interrupción del ADC – ADC\_IRQHandler – para que cada vez que interrumpiera el programa, leyéramos el valor del ADC, lo transformásemos de Voltios a Distancia y este valor lo escribiéramos en el código html en el lugar correspondiente.

En depuración observamos que el programa no entraba en la función de interrupción del ADC incluso cuando la función del SysTick así se lo señalaba por lo que cambiamos la forma de interrumpir del ADC y la establecimos de modo que interrumpiera de forma automática con la llegada del Match 1 del Timer 0.

26:24	START	When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
000		No start (this value should be used when clearing PDN to 0).	
001		Start conversion now.	
010		Start conversion when the edge selected by bit 27 occurs on the P2.10 / EINT0 / NMI pin.	
011		Start conversion when the edge selected by bit 27 occurs on the P1.27 / CLKOUT / USB_OVRCRn / CAP0.1 pin.	
100		Start conversion when the edge selected by bit 27 occurs on MAT0.1. Note that this does not require that the MAT0.1 function appear on a device pin.	
101		Start conversion when the edge selected by bit 27 occurs on MAT0.3. Note that it is not possible to cause the MAT0.3 function to appear on a device pin.	
110		Start conversion when the edge selected by bit 27 occurs on MAT1.0. Note that this does not require that the MAT1.0 function appear on a device pin.	
111		Start conversion when the edge selected by bit 27 occurs on MAT1.1. Note that this does not require that the MAT1.1 function appear on a device pin.	

Figura 49. Imagen con extracto del manual que contiene parte del registro ADCR del ADC

De este modo al depurar el programa veíamos como ahora sí entraba en la interrupción y escribía en el código html de forma correcta. Sin embargo, a pesar de las modificaciones, al ejecutar el programa, éste no mostraba nada por la Web, por lo que dedujimos que el problema residía en el hecho de que los datos había que escribirlos en el código html de forma conjunta, y no un dato cada vez que interrumpía el ADC.

Para cambiar esto, definimos un array de punteros, que durante la primera ejecución fijan su posición, es decir, el primer puntero del array apunta a D1 y así sucesivamente.

```
unsigned char *pData[4]; //Array de punteros que apuntan hacia el código html
```

```
if ((*Key == 'D') && (*(Key + 2) == '§')) //Hemos encontrado el lugar donde escribir algún dato
{
    if (*(Key+1) == '1')
    {
        pData[0]=Key; //Ponemos al puntero pData apuntando al mismo sitio que Key (a D1)
    }
    if (*(Key+1) == '2')
    {
        pData[1]=Key; //Apunta a D2
    }
    if (*(Key+1) == '3')
    {
        pData[2]=Key; //Apunta a D3
    }
    if (*(Key+1) == '4')
    {
        pData[3]=Key; //Apunta a D4
    }
}
```

Figura 50. Imágenes que detallan la definición de los punteros

Finalmente, y tras solucionar un pequeño problema según el cual el programa se detenía debido a que entraba en la interrupción del ADC antes de que se hubieran fijado los punteros y esto producía un error en la simulación, después de aplicar los cambios

mencionados conseguimos obtener los resultados esperados tal y como muestran algunas imágenes a continuación.

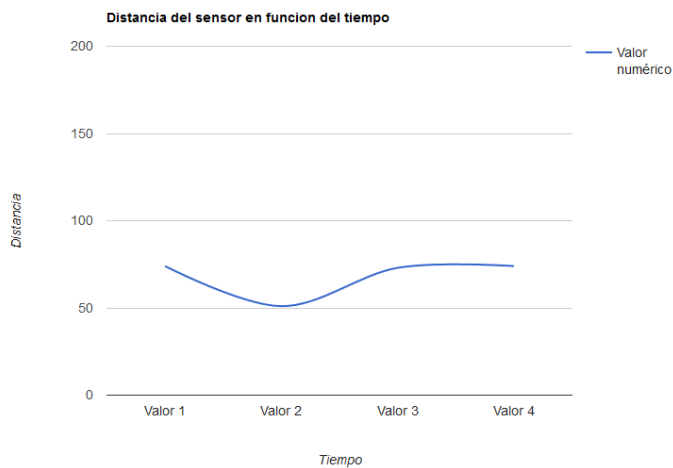


Page Count: 24

Figura 51. Gráfico de la aplicación sin obstáculos

En esta imagen observamos como el valor de la distancia se mantiene más o menos constante en ausencia de objetos durante los 4 valores de los que dispone esta simulación.





Page Count: 13

Figura 52. Gráfico de la aplicación con obstáculos

En esta figura vemos como oscila el valor de la distancia cuando pasamos brevemente la mano a una distancia considerable del sensor.

## 11. Conclusiones

Como hemos mostrado anteriormente, hemos logrado desarrollar una funcionalidad basada en javascript que proyecta un gráfico sobre la interfaz de la página web, mostrando el valor de la distancia que el sensor de distancia nos transmite cada cierto intervalo de tiempo.

Esto demuestra la viabilidad de utilizar un microcontrolador como servidor de una página web, a la vez que abre un amplio abanico de posibilidades de investigación de nuevas funcionalidades de este tipo.

Teniendo en cuenta lo expuesto anteriormente, podemos afirmar que el proyecto ha sido realizado con éxito ya que se han logrado los objetivos propuestos que consistían en analizar el código proporcionado y desarrollar nuevas funcionalidades.

En un futuro, se podrían implementar funcionalidades más avanzadas e incluso la posibilidad de conseguir que la conexión funcione en sentido inverso, es decir, que desde la página web el usuario sea capaz de modificar valores de la ejecución en tiempo real tales como el tiempo de intervalo o el número de muestras por gráfico.

Por todo esto considero que este proyecto ha sido de mucha ayuda en mi preparación como ingeniero ya que precisa de conocimientos en diversas materias que me han obligado a investigar y aprender una serie de recursos que a buen seguro me serán muy útiles en mi vida laboral.

## 12. Planificación temporal

A continuación, vamos a exponer la forma en la que se ha estructurado el proyecto y una aproximación del tiempo que ha consumido cada etapa:

- Estructuración del proyecto: En esta primera etapa, se planteó como se iba a abordar el proyecto y se elaboró un plan de acción para llevarlo a cabo. Su duración fue de aproximadamente **1 semana**.
- Documentación: Una vez se tuvo consciencia de que había que trabajar con un lenguaje nuevo (HTML), se comenzó la búsqueda de documentación e información acerca de este lenguaje para ser capaces de desarrollar la funcionalidad deseada en él. **2 semanas**.
- Análisis del código proporcionado: Como ya hemos explicado, easyweb nos proporcionó un programa para comunicar el microcontrolador con la página web, que tuvimos que analizar en detalle para comprender todas las consideraciones necesarias ya que influirían en la forma de desarrollar nuestra funcionalidad. **3 semanas**.
- Primera etapa de programación: Distinguimos la programación en 2 etapas debido a que se trata de dos partes claramente diferenciadas. En esta primera, predominó la búsqueda en el manual del LPC1768 para encontrar comandos que se adaptaran a lo que necesitábamos, así como solución de errores de compilación y cambios en la estructura del programa. **3 semanas**.
- Segunda etapa de programación: Esta etapa se caracteriza por, una vez obtenida una versión – aunque inmadura – de la funcionalidad a implementar, destaca sobremanera las repetidas depuraciones del código y la depuración de errores de programación y fallos en el diseño. **4 semanas**.
- Personalización gráfica: Una vez obtenida la funcionalidad deseada funcionando de forma correcta, comenzamos a explorar las opciones que nos ofrecía el javascript de Google para modificar infinidad de parámetros relacionados con el gráfico. Esta etapa duró **2 semanas**.

- Redacción de la memoria: Aunque ya se había empezado a redactar tiempo atrás, el avance más importante en este apartado se logró a partir de la consecución de la segunda parte de programación, ya que gracias a esto disponíamos de toda la información necesaria para completar los apartados. **3 semanas.**
- Entrega y revisión de la memoria: Una vez obtenida una versión más o menos definitiva de la memoria, se procedió a mostrársela al tutor y pulirla. Esta acción llevo **1 semana.**
- Defensa del proyecto: Al haber entregado la memoria, el único paso restante es el de defender el trabajo realizado, si es posible mediante una muestra en tiempo real de la funcionalidad desarrollada. **1semana.**

[illegible]

## 13. Bibliografía

- [1] <http://www.fotonostra.com/digital/paginasweb.htm>
- [2] <http://www.masadelante.com/faqs/servidor>
- [3] <http://www.electronicaestudio.com/microcontrolador.htm>
- [4] <http://www.monografias.com/trabajos12/microco/microco.shtml>
- [5] <http://www.hotmcu.com/lpc1768minidk2-development-board-p-55.html>
- [6] <http://www2.keil.com/mdk5/uvision/>
- [7] [https://www.embeddedartists.com/products/acc/acc\\_jtag\\_adapter\\_kit.php](https://www.embeddedartists.com/products/acc/acc_jtag_adapter_kit.php)
- [8] <http://tecnologia-facil.com/que-es/que-es-un-cable-ethernet/>
- [9] <http://www.redeszone.net/redes/que-cable-de-red-ethernet-debo-utilizar-guia-de-eleccion-para-categoria-5-5e-6-y-6a/>
- [10] [https://engineering.purdue.edu/ME588/SpecSheets/sharp\\_gp2d12.pdf](https://engineering.purdue.edu/ME588/SpecSheets/sharp_gp2d12.pdf)
- [11] Libro: El lenguaje de programación C (Brian W. Kernigan y Dennis M. Ritchie) Ed Parson Education
- [12] Libro: HTML and CSS: Design and Build Websites (Jon Duckett) Ed Wiley
- [13] [http://www.nxp.com/documents/user\\_manual/UM10360.pdf](http://www.nxp.com/documents/user_manual/UM10360.pdf)

## 14. Anexos

### 14.1 Anexo 1. Código Easyweb original

```

/*****
*****
***** Name: easyweb.c *****
***** Ver.: 1.0 *****
***** Date: 07/05/2001 *****
***** Auth: Andreas Dannenberg *****
***** HTWK Leipzig *****
***** university of applied sciences *****
***** Germany *****
***** Func: implements a dynamic HTTP-server by using *****
***** the easyWEB-API *****
*****
*****/

```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include "LPC17xx.h" // Keil: Register definition file for LPC17xx
```

```
extern uint32_t SystemFrequency;
```

```
#define extern // Keil: Line added for modular project management
```

```
#include "easyweb.h"
```

```
#include "type.h"
```

```
#include "EMAC.h"      // Keil: *.c -> *.h  // ethernet packet driver
#include "tcpip.h"      // Keil: *.c -> *.h  // easyWEB TCP/IP stack

#include "webpage.h"          // webside for our HTTP server (HTML)

unsigned int pagecounter = 0;
unsigned int adcValue  = 0;

extern void TCPClockHandler(void);

volatile DWORD TimeTick = 0;

/* SysTick interrupt happens every 10 ms */
void SysTick_Handler (void) {
    TimeTick++;
    if (TimeTick >= 20) {
        TimeTick = 0;
        // LPC_GPIO3->FIOPIN ^= 1 << 25; //CAMBIADO!
        LPC_GPIO2->FIOPIN ^= 1 << 0; //CAMBIADO!
        TCPClockHandler();
    }
}

//void main(void)
int main(void)
{
    //InitOsc();      // Keil: No oscillator initialization necessary at this time.
    //InitPorts();    // Keil: No port initialization necessary at this time.
```



```

SystemInit();                      /* setup core clocks */
SysTick_Config(SystemFrequency/100); /* Generate interrupt every 10 ms */

LPC_GPIO0->FIODIR |= 1 << 21;
LPC_GPIO0->FIOPIN |= 1 << 21;

// LPC_GPIO3->FIODIR |= 1 << 25;          /* P2.0 defined as Output (LED)
CAMBIADO! */
LPC_GPIO2->FIODIR |= 1 << 0;
LPC_PINCON->PINSEL3 |= (3u<<30);          /* P1.31 is AD0.5 */
LPC_SC->PCONP |= (1<<12);                  /* Enable power to ADC block */
LPC_ADC->ADCR = (1<< 5) |                  /* select AD0.5 pin */
                (4<< 8) |                  /* ADC clock is 25MHz/5 */
                (1<<21);                  /* enable ADC */

TCPLowLevelInit();

/*
*(unsigned char *)RemoteIP = 24;          // uncomment those lines to get the
*((unsigned char *)RemoteIP + 1) = 8;      // quote of the day from a real
*((unsigned char *)RemoteIP + 2) = 69;     // internet server! (gateway must be
*((unsigned char *)RemoteIP + 3) = 7;      // set to your LAN-router)

TCPLocalPort = 2025;
TCPRemotePort = TCP_PORT_QOTD;

TCPActiveOpen();

while (SocketStatus & SOCK_ACTIVE)        // read the quote from memory
{
    // by using the hardware-debugger

```

```

    DoNetworkStuff();
}
*/

HTTPStatus = 0;                // clear HTTP-server's flag register

TCPLocalPort = TCP_PORT_HTTP;    // set port we want to listen to

while (1)                        // repeat forever
{
    if (!(SocketStatus & SOCK_ACTIVE)) TCPPassiveOpen(); // listen for incoming TCP-
connection
    DoNetworkStuff();            // handle network and easyWEB-stack
                                // events

    HTTPServer();
}
}

```

```

// This function implements a very simple dynamic HTTP-server.
// It waits until connected, then sends a HTTP-header and the
// HTML-code stored in memory. Before sending, it replaces
// some special strings with dynamic values.
// NOTE: For strings crossing page boundaries, replacing will
// not work. In this case, simply add some extra lines
// (e.g. CR and LFs) to the HTML-code.

```

```

void HTTPServer(void)
{
    if (SocketStatus & SOCK_CONNECTED)    // check if somebody has connected to
our TCP

```

```

{
    if (SocketStatus & SOCK_DATA_AVAILABLE)    // check if remote TCP sent data
        TCPReleaseRxBuffer();                // and throw it away

    if (SocketStatus & SOCK_TX_BUF_RELEASED)    // check if buffer is free for TX
    {
        if (!(HTTPStatus & HTTP_SEND_PAGE))    // init byte-counter and pointer to
        website
        {
            // if called the 1st time
            HTTPBytesToSend = sizeof(WebSide) - 1; // get HTML length, ignore trailing zero
            PWebSide = (unsigned char *)WebSide; // pointer to HTML-code
        }

        if (HTTPBytesToSend > MAX_TCP_TX_DATA_SIZE) // transmit a segment of
        MAX_SIZE
        {
            if (!(HTTPStatus & HTTP_SEND_PAGE))    // 1st time, include HTTP-header
            {
                memcpy(TCP_TX_BUF, GetResponse, sizeof(GetResponse) - 1);
                memcpy(TCP_TX_BUF + sizeof(GetResponse) - 1, PWebSide,
                MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1);
                HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
                PWebSide += MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
            }
            else
            {
                memcpy(TCP_TX_BUF, PWebSide, MAX_TCP_TX_DATA_SIZE);
                HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE;
                PWebSide += MAX_TCP_TX_DATA_SIZE;
            }
        }
    }
}

```

```

    TCPTxDataCount = MAX_TCP_TX_DATA_SIZE; // bytes to xfer
    InsertDynamicValues();                // exchange some strings...
    TCPTransmitTxBuffer();                // xfer buffer
}
else if (HTTPBytesToSend)                // transmit leftover bytes
{
    memcpy(TCP_TX_BUF, PWebSide, HTTPBytesToSend);
    TCPTxDataCount = HTTPBytesToSend;    // bytes to xfer
    InsertDynamicValues();                // exchange some strings...
    TCPTransmitTxBuffer();                // send last segment
    TCPClose();                          // and close connection
    HTTPBytesToSend = 0;                  // all data sent
}

HTTPStatus |= HTTP_SEND_PAGE;           // ok, 1st loop executed
}
}
else
    HTTPStatus &= ~HTTP_SEND_PAGE;       // reset help-flag if not connected
}

// searches the TX-buffer for special strings and replaces them
// with dynamic values (AD-converter results)

```

```

void InsertDynamicValues(void)
{
    unsigned char *Key;
    char NewKey[5];

```

```

unsigned int i;

if (TCPTxDataCount < 4) return;           // there can't be any special string

Key = TCP_TX_BUF;

for (i = 0; i < (TCPTxDataCount - 3); i++)
{
    if (*Key == 'A')
        if (*(Key + 1) == 'D')
            if (*(Key + 3) == '%')
                switch (*(Key + 2))
                {
                    case '8' :           // "AD8%"?
                    {
                        adcValue = rand()%4024;           // get AD value
                        sprintf(NewKey, "0x%03X", adcValue); // insert AD converter value
                        memcpy(Key, NewKey, 5);
                        break;
                    }
                    case '7' :           // "AD7%"?
                    {
                        sprintf(NewKey, "%3u", (adcValue*100)/4024); // copy saved value from
previous read
                        memcpy(Key, NewKey, 3);
                        break;
                    }
                    case '1' :           // "AD1%"?
                    {

```

```

        sprintf(NewKey, "%3u", ++pagecounter); // increment and
insert page counter
        memcpy(Key, NewKey, 3);
        *(Key + 3) = ' ';

        break;
    }
}

Key++;
}
}

```

## 14.2 Anexo 2. Código Easyweb definitivo

```

/*****
*****

***** Name: easyweb.c *****
***** Ver.: 1.0 *****
***** Date: 07/05/2001 *****
***** Auth: Andreas Dannenberg *****
***** HTWK Leipzig *****
***** university of applied sciences *****
***** Germany *****
***** Func: implements a dynamic HTTP-server by using *****
***** the easyWEB-API *****
*****

*****/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

```

```
#include "LPC17xx.h"    // Keil: Register definition file for LPC17xx

extern uint32_t SystemFrequency;

#define extern          // Keil: Line added for modular project management
#define Fpclk 25e6
#include "easyweb.h"

#include "type.h"
#include "EMAC.h"        // Keil: *.c -> *.h    // ethernet packet driver
#include "tcpip.h"       // Keil: *.c -> *.h    // easyWEB TCP/IP stack

#include "webpage.h"          // webside for our HTTP server (HTML)

unsigned int pagecounter = 0;
unsigned int adcValue  = 0;
unsigned int convertido = 0;
unsigned int distanciaobstaculo = 0;
unsigned char *Key;
char NewKey[5];
float refresco = 2;
float nmuestras = 4; //Cambiar estas dos variables mediante código html
unsigned char *pData[4];
char NewpData[3];
int arraydistancias[4]={10,10,10,10};
unsigned int conversion=0;
unsigned int cuentaSystick =0;
unsigned int j=0;
```

```

extern void TCPClockHandler(void);

volatile DWORD TimeTick = 0;

/* SysTick interrupt happens every 10 ms */
void SysTick_Handler (void) {
    TimeTick++;
    if (TimeTick >=20) {
        TimeTick=0;
        LPC_GPIO2->FIOPIN ^= 1 << 0;
        cuentaSystick++;
        TCPClockHandler();
    }
    if (cuentaSystick>=2) {
        cuentaSystick=0;
        //LPC_ADC->ADCR |=(1<<24);
    }
}

void ADC_IRQHandler(void)
{
    float voltios;

    voltios= ((LPC_ADC->ADGDR >>4)&0xFFF)*3.3/4095;    // se borra automat.
    el flag DONE al leer ADCGDR

    distanciaobstaculo = 23.75/(voltios-0.2)-0.42; //Algoritmo del sensor
    arraydistancias[convertido]=distanciaobstaculo;
    convertido++;
}

```



```

//void main(void)

int main(void)
{
//InitOsc();          // Keil: No oscillator initialization necessary at this time.
//InitPorts();        // Keil: No port initialization necessary at this time.

SystemInit();          /* setup core clocks */
SysTick_Config(SystemFrequency/100);    /* Generate interrupt every 10 ms */

LPC_GPIO0->FIODIR |= 1 << 21;
LPC_GPIO0->FIOPIN |= 1 << 21;

LPC_GPIO2->FIODIR |= 1 << 0;          /* P2.0 defined as Output (LED) */

LPC_PINCON->PINSEL3 |= (3u<<30);      /* P1.31 is AD0.5 */
LPC_SC->PCONP |= (1<<12);              /* Enable power to ADC block */
LPC_ADC->ADCR = (1<< 5) |              /* select AD0.5 pin */
               (4<< 8) |               /* ADC clock is 25MHz/5 */
               (1<<21) |               /* enable ADC */
               (6<<24);                // Inicio de conversión cuando escriba la
siguiente instruccion

//LPC_ADC->ADCR(1<<24);

////////////////////////Código introducido por mi////////////////////////

LPC_ADC->ADINTEN= (1<<0)|(1<<8);        // Hab. interrupción fin de
conversión canal 0

NVIC_EnableIRQ(ADC_IRQn);              // Habilita
interrupcion

NVIC_SetPriority(ADC_IRQn,1);           // Prioridad 1

//////////////////TIMER1//////////////////

```

```

        LPC_SC->PCONP|=(1<<2);                                //
Power on
        LPC_TIM1->PR = 0x00;                                    // Prescaler a 0
        LPC_TIM1->MCR |= (1<<1);                                // Reset del contador MR0
        LPC_TIM1->MR0 = (Fpclk*(refresco/(nmuestras+1)))-1; // Hacemos que la el el
Timer 0 llegue al Match nmuestras veces a lo largo de refresco segundos
        LPC_TIM1->EMR|=(3<<4);
        LPC_TIM1->TCR = 0x01; //Se habilita el timer para contar
////////////////////////////////////
TCPLowLevelInit();

/*
*(unsigned char *)RemoteIP = 24;        // uncomment those lines to get the
*((unsigned char *)RemoteIP + 1) = 8;    // quote of the day from a real
*((unsigned char *)RemoteIP + 2) = 69;    // internet server! (gateway must be
*((unsigned char *)RemoteIP + 3) = 7;    // set to your LAN-router)

TCPLocalPort = 2025;
TCPRemotePort = TCP_PORT_QOTD;

TCPActiveOpen();

while (SocketStatus & SOCK_ACTIVE)      // read the quote from memory
{
    // by using the hardware-debugger
    DoNetworkStuff();
}
*/

HTTPStatus = 0;                        // clear HTTP-server's flag register

```

```

TCPLocalPort = TCP_PORT_HTTP;           // set port we want to listen to

while (1)                                // repeat forever
{
    if (!(SocketStatus & SOCK_ACTIVE)) TCPPassiveOpen(); // listen for incoming TCP-
connection
    DoNetworkStuff();                     // handle network and easyWEB-stack
                                         // events
    HTTPServer();
}
}

```

// This function implements a very simple dynamic HTTP-server.

// It waits until connected, then sends a HTTP-header and the

// HTML-code stored in memory. Before sending, it replaces

// some special strings with dynamic values.

// NOTE: For strings crossing page boundaries, replacing will

// not work. In this case, simply add some extra lines

// (e.g. CR and LFs) to the HTML-code.

```

void HTTPServer(void)
{
    if (SocketStatus & SOCK_CONNECTED)    // check if somebody has connected to
our TCP
    {
        if (SocketStatus & SOCK_DATA_AVAILABLE) // check if remote TCP sent data
        {
            TCPReleaseRxBuffer();             // and throw it away

            if (SocketStatus & SOCK_TX_BUF_RELEASED) // check if buffer is free for TX
            {

```

```

    if (!(HTTPStatus & HTTP_SEND_PAGE))    // init byte-counter and pointer to
    webside

    {
        // if called the 1st time
        HTTPBytesToSend = sizeof(WebSide) - 1; // get HTML length, ignore trailing zero
        PWebSide = (unsigned char *)WebSide; // pointer to HTML-code
    }

    if (HTTPBytesToSend > MAX_TCP_TX_DATA_SIZE) // transmit a segment of
    MAX_SIZE
    {
        if (!(HTTPStatus & HTTP_SEND_PAGE))    // 1st time, include HTTP-header
        {
            memcpy(TCP_TX_BUF, GetResponse, sizeof(GetResponse) - 1);
            memcpy(TCP_TX_BUF + sizeof(GetResponse) - 1, PWebSide,
            MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1);
            HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
            PWebSide += MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
        }
        else
        {
            memcpy(TCP_TX_BUF, PWebSide, MAX_TCP_TX_DATA_SIZE);
            HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE;
            PWebSide += MAX_TCP_TX_DATA_SIZE;
        }
    }

    TCPTxDataCount = MAX_TCP_TX_DATA_SIZE; // bytes to xfer
    InsertDynamicValues();    // exchange some strings...
    TCPTransmitTxBuffer();    // xfer buffer
}

else if (HTTPBytesToSend)    // transmit leftover bytes
{

```

```

    memcpy(TCP_TX_BUF, PWebSide, HTTPBytesToSend);
    TCPTxDataCount = HTTPBytesToSend;    // bytes to xfer
    InsertDynamicValues();                // exchange some strings...
    TCPTransmitTxBuffer();                // send last segment
    TCPClose();                          // and close connection
    HTTPBytesToSend = 0;                  // all data sent
}

    HTTPStatus |= HTTP_SEND_PAGE;        // ok, 1st loop executed
}
}
else
    HTTPStatus &= ~HTTP_SEND_PAGE;       // reset help-flag if not connected
}

// searches the TX-buffer for special strings and replaces them
// with dynamic values (AD-converter results)

void InsertDynamicValues(void)
{

    unsigned int i;

    if (TCPTxDataCount < 4) return;       // there can't be any special string

    Key = TCP_TX_BUF;

    for (i = 0; i < (TCPTxDataCount - 3); i++)

```

```

{
    if (*Key == 'A')
        if (*(Key + 1) == 'D')
            if (*(Key + 3) == '%')
                switch (*(Key + 2))
                {
                    case '8' :                // "AD8%"?
                    {
                        adcValue = rand()%4024;        // get AD value
                        sprintf(NewKey, "0x%03X", adcValue);    // insert AD converter value
                        memcpy(Key, NewKey, 5);
                        break;
                    }
                    case '7' :                // "AD7%"?
                    {
                        sprintf(NewKey, "%3u", (adcValue*100)/4024);    // copy saved value from
previous read
                        memcpy(Key, NewKey, 3);
                        break;
                    }
                    case '1' :                // "AD1%"?
                    {
                        sprintf(NewKey, "%3u", ++pagecounter);    // increment and insert page
counter
                        memcpy(Key, NewKey, 3);
                        *(Key + 3) = ' ';
                        break;
                    }
                }
}

```

////////////////////////////////////

```

        if ((*Key == 'D') && (*(Key + 2) == '%')) // Hemos encontrado el lugar
donde escribir algun dato
    {
        if (*(Key + 1) == '1')
        {
            pData[0] = Key; // Ponemos al
puntero pData apuntando al mismo sitio que Key (a D1)
            // A cada Match se inicia la
conversion y el reloj se resetea para seguir contando.
            // while (convertido = 0); // bucle para
que espere a que termine la conversion
        }
        if (*(Key + 1) == '2')
        {
            pData[1] = Key;
        }
        if (*(Key + 1) == '3')
        {
            pData[2] = Key;
        }
        if (*(Key + 1) == '4')
        {
            pData[3] = Key;
            convertido = 0; // Se resetea la
variable
            for (j = 0; j < 4; j++) {
                sprintf(NewpData, "%3u",
(arraydistancias[j]));

                memcpy(pData[j], NewpData, 3);
            }

```

```
        }  
    }  
  
    Key++;  
  
}  
}
```